Third Annual Report for Perception for Outdoor Navigation

Principal Investigators:
Charles Thorpe and Takeo Kanade
Other Faculty:
Martial Hebert and Dean Pomerleau

CMU-RI-TR-92-16

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

September 1992

This report reviews progress at Carnegie Mellon from August 16, 1991 to August 15, 1992 on research sponsored by DARPA, DoD, monitored by the U.S. Army Topographic Engineering Center under contract DACA 76-89-C-0014, titled "Perception for Outdoor Navigation." Parts of this work were also supported by a Hughes Fellowship, by a DARPA Research Assistantship in Parallel Processing administered by the Institute for Advanced Computer Studies of the University of Maryland, and by NSF, under NSF Contract BCS-9120655, titled "Annotated Maps for Autonomous Underwater Vehicles."

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1992	technical

4. TITLE AND SUBTITLE

Third Annual Report for Perception for Outdoor Navigation

5. FUNDING NUMBERS

DACA 76-89-C-0014

6. AUTHOR(S)

Principal Investigators: Charles Thorpe and Takeo Kanade
Other Faculty: Martial Hebert and Dean Pomerleau

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

8. PERFORMING ORGANIZATION REPORT NUMBER

CMU-RI-TR-92-16

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

DARPA

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release;
Distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This report reviews progress at Carnegie Mellon from August 16, 1991 to August 15, 1992 on research sponsored by DARPA, DoD, monitored by the U.S. Army Topographic Engineering Center under contract DACA 76-89-C-0014, titled "Perception for Outdoor Navigation."

Research supported by this contract includes perception for road following, terrain mapping for off-road navigation, and systems softwre for building integrated mobile robots. We overview our efforts for the year, and list our publications and personnel, then provide further detail on several of our subprojects.

14. SUBJECT TERMS

15. NUMBER OF PAGES
63 pp

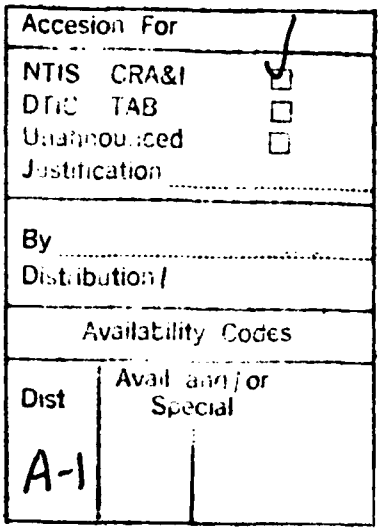
16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
unlimited	unlimited	unlimited	unlimited

Contents

List of Figures

List of Tables



Abstract

This report reviews progess at Carnegie Mellon from August 16, 1991 to August 15, 1992 on research sponsored by DARPA, DOD, monitored by the US Army Topographic Engineering Center under contract DACA 76-89-C-0014. titled "Perception for Outdoor Navigation".

Research supported by this contract includes perception for road following, terrain mapping for off-road navigation, and systems software for building integrated mobile robots. We overview our efforts for the year, and list our publications and personnnel, then provide further detail on several of our subprojects.

Chapter 1

Introduction

This report reviews progress at Carnegie Mellon from August 16,1991 to August 15, 1992 on research sponsored by DARPA, DOD, monitored by the US Army Topographic Engineering Center under contract DACA 76-89-C-0014. titled "Perception for Outdoor Navigation". Parts of this work were also supported by a Hughes Fellowship, by a DARPA Research Assistantship in Parallel Processing administered by the Institute for Advanced Computer Studies of the University of Maryland, and by NSF, under NSF Contract BCS-9120655, titled "Annotated Maps for Autonomous Underwater Vehicles".

During this third year of the contract, we have made continued progress in the area of machine perception for outdoor mobile robots. We have built new algorithms (in sonar-based navigation, neural networks, and range data analysis); we have built new systems that handle uncertainty, and have incorporated those systems into our annotated maps; and we have been involved in a number of programmatic activities, including two thesis defenses and playing a pivotal role in the development of the DARPA program in Unmanned Ground Vehicles.

The report begins with a summary of the year's activities and accomplishments, in this chapter. Chapter 2, "Sonar-Based Outdoor Vehicle Navigation and Obstacle Avoidance", provides more details on the sonar-based map building and driving. Chapter 3, "Evaluation of 3-D Measurements From Imaging Laser Radars" reports on extensive experiments conducted with our two laser range finders. Chapter 4, "Progress in Neural Network-Based Vision for Autonomous Robot Driving" reports in detail on our continued progress in autonomous on-road driving using simulated neural nets. Finally, the last chapter, "Integrating Position Measurement and Image Understanding for Autonomous Vehicle Navigation", provides a complete description of the design and implementation of the uncertainty filtering algorithms that we have integrated with our navigation and perception modules.

1.1 Algorithms and Modules

Integrated System for 3D Range Sensing

The focus of this year's 3D sensing research has been integration. GANESHA is our new occupancy grid representation for 3D objects. It has proven useful for integrating the range data collected by multiple sensors, including a sonar array, a microwave radar sensor, and the ERIM scanning laser sensor. In this representation, when a 3D object is detected by one or more sensors, the grid cell(s) corresponding to the object's position are marked as occupied, with an annotation indicating which sensor(s) detected the object. If the object is detected more than once by the same sensor or by multiple sensors, the mapping system increases a confidence value indicating the likelihood that the grid

cell is occupied. Chapter 2 describes the grid representation, and the results achieved using it to detect and avoid obstacles, and to track continuous objects in the environment such as guard rails and rows of parked cars. Recently, we have made preliminary experiments in integrating range information provided by our stereo vision system into the occupancy grid representation.

In addition to merging data from multiple sensors, work this year in 3D range sensing has also addressed integrating range sensing with other processing modules in structured environments. The highlight of this effort was a demonstration that combined neural network-based vision for road following, sonar-based range sensing for obstacle avoidance and following parallel to rows of parked cars, and ERIM-based landmark recognition. The combined system was able to repeatedly drive around a busy parking lot, stop at stop signs and avoid dynamic obstacles. We have also begun preliminary experiments in using the grid-based object representation for automatic parallel parking.

Laser Range Finder Data: Evaluation and Processing

We have been using range data from a laser range finder to find obstacles, to model terrain, and to recognize landmarks. Obstacle detection is now an integral part of our navigation systems. We have also integrated in our systems the mapping and matching of simple landmarks that can be represented by their locations. We are extending this work to mapping and matching landmarks using more complete shape descriptions. The shape descriptions are built for each object by using the range images collected as the vehicle travels. Once integrated in the annotated map system, the landmark matching capability makes map-based navigation in complex environments possible by providing an effective way of discriminating between landmarks, and an efficient way to compute vehicle pose relative to landmarks. We have demonstrated the object modeling and matching using the Erim laser range finder on both Navlab I and Navlab II. We are integrating those capabilities into our navigation systems.

In addition to research in range data processing, we have been investigating issues related to sensor performance. Fast and reliable range data processing is critical for autonomous driving, both off-road and on-road. Our autonomous driving performance is currently limited by the available range finding sensors. In order to quantify those limitations, and to better understand the performance of our algorithms, we have performed a series of experiments. The experiments involve a detailed analysis of the performance of the scanner, adn the algorithms that process the scanner's data. They provide us with useful information such as the size of detectable objects as a function of their range from the sensor. The results from these experiments allow us to quantify the performance of our algorithms, and suggest methods of improving them. The experiments also provide a baseline to guide the design of the next generation of range sensors. For the mobile robot community, these experiments provide a reference benchmark on the performance of range data analysis for navigation. As a related activity, we are starting to evaluate the use of passive stereo data in our algorithms.

Further information on laser range finder evaluation and performance can be found in Chapter 3.

Neural Networks for Autonomous Driving

The ALVINN neural network based driving system has achieved a number of milestones this year, including a new speed record of 55 mph, and distance record of 21.2 miles of continuous autonomous driving. In order to achieve these advances, a number algorithmic improvements to the ALVINN system have been required. One of the most important is a new technique which allows ALVINN to learn to drive accurately from a very limited amount of training data. Since ALVINN learns by observing only few minutes of human driving performance, there are driving situations that ALVINN is not exposed to during training, such as overtaking another car. Using simple symbolic models of how these rare situations will appear, and then augmenting the training data with artificial examples of these situations,

ALVINN learns to handle them without problem. The details of this technique, and its impact on performance, are described in Chapter 4.

Another major effort this year in the ALVINN system has been in the area of quantitative performance analysis. These techniques have allowed us to measure how various improvements to the ALVINN system impact driving performance, and to compare ALVINN's driving with that of people. We have found that ALVINN's driving accuracy, as measured by how close to the center of the road the system keeps the vehicle, is comparable to that of a human driver. In addition, the quantitative models we have developed for ALVINN's driving performance have allowed us to integrate ALVINN with other autonomous navigation methods, as described in Chapter 5.

Systems

In previous years, we have developed several systems integrating map building, road following, and range data processing. The perception modules in those systems are tied together through the vehicle map tracker, the GANESHA local map maintainer, Annotated Maps, and the DARN integration architecture. During the year covered by this report, we have extended those systems to include sonar and radar processing, and to carry out more complex missions. In particular, we have demonstrated the flexibility of our systems by including new perception modules without substantial modifications of the core system. We have also demonstrated the performance of the system by demonstrating it in mission scenarios that require up to twenty different modules and that require complex interactions between perception modules controlled by the annotated map system. These interactions included, for example, switching from one neural network to another in the course of a mission, or switching between different obstacle detection modes.

In addition to building more complex systems, our emphasis this year has been on building more robust systems through the addition of uncertainty filtering. The need for uncertainty filtering arises because all the measurements used for driving the vehicle, both from internal position sensors and from perception sensors, are corrupted by noise. This is not a problem for short runs, since the errors do not build up to significant levels. However, as soon as missions become longer and more complex, the accumulation of errors leads to rapid performance degradation. For these reasons, we have developed a mechanism for combining position estimates which explicitly takes into account the uncertainty associated with each position measurement. The sources of position estimates are the estimate from INS and dead reckoning, the position estimates obtained by matching objects detected in range images with previously mapped objects, and lateral position of the vehicle with respect to the center of the road estimated by ALVINN. The uncertainty values are combined using the recursive Kalman filter formulation. The Kalman filter provides a clean and uniform approach. The approach is sensor-independent in that position measurements from new sensors can be integrated without modifying the core of the position filtering system. The system continuously updates the optimal estimate of vehicle position and its uncertainty from all available measurements. We have demonstrated the performance of the uncertainty filtering in a system that involved driving on-road and updating position along the way using landmarks. We expect this uncertainty filtering to be part of all future systems.

Details of the theory and implementation of uncertainty filtering are given in Chapter 5.

Other work in progress

YARF

Our model-based road follower is YARF (Yet Another Road Follower). During the past year, YARF has expanded in several ways. The individual trackers in YARF output candidate positions for road features such as painted lines or the pavement edge. In early implementations, the road model was fit to these tracked features with a least-squares method.

This year, we have demonstrated using a least-median-of-squares (LMS) method, from the domain of robust statistics, to fit the road model to the data. Using this technique, we have successfully detected the divergence of road features at a highway exit, then rejected the exit and continued to track the main road. Final experiments are now underway for YARF to drive the Navlab through street intersections, using map information and LMS fitting. The design of YARF has been reported previously; final results are forthcoming in Karl Kluge's thesis.

MANIAC

An open problem with ALVINN is automatically handling a variety of roads. MANIAC (Multiple Alvinn Networks In Autonomous Control) is an innovative approach to that problem. MANIAC starts by training several individual ALVINN nets, for example one for dirt roads, one for narrow paved trails, and one for four-lane roads. Then, in a separate training phase, a new net is built that listens to the individual ALVINNs and outputs the best steering direction. Our hypothesis is that MANIAC should not simply select among the nets, but should use information from all of the individual ALVINNs. The rationale is that although a net trained on a different type of road may be wrong, it might be incorrect in a consistent way, which could still be a useful cue. For instance, if the road width is different from the road on which a particular net has been trained, the net might output two steering directions, one too far to the left and one too far to the right. The MANIAC net, seeing that double-valued output, should learn to steer between the two steering directions. We have tried MANIAC architectures that look at the outputs of the individual ALVINNs, and others that look at the hidden units. Our results to date are promising but not yet conclusive. This is an excellent example of cooperative research. The first stages of MANIAC were built by our student, Todd Jochem, while in residence at Martin Marietta as part of his DARPA Research Assistantship on Parallel Processing. First results on MANIAC will appear in this February's IAS-3 conference.

RACCOON

We have demonstrated tail light tracking at night with the RACCOON system (Real-time Autonomous Car Chaser Operating Optimally at Night). RACCOON tracks the bounding box of the tail lights of a car, running at about 10 Hz on a Sun workstation. During a live run, RACCOON runs on the Navlab II and generates steering commands. Rather than simply servoing to point at the lights of the lead vehicle, we infer the range and bearing to the lead vehicle, record the location in world coordinates on the ground, and feed those recorded points to our trajectory generator. This keeps us from clipping corners, and allows us to keep a known distance from the lead vehicle. We have demonstrated vehicle tracking at speeds up to 20 mph. We have implemented simple recovery strategies, so if we lose a the lead car due to momentary occlusion we can often recover and continue the run.

1.2 Personnel

Theses defended:

- Dean Pomerleau, "Neural Network Perception for Mobile Robot Guidance", February 1992.
- Douglas Reece, "Selective Perception for Robot Driving", May 1992.

New faculty:

- Dean Pomerleau, neural networks, active vision

New PhD students:

- Charalambos Athanassiou, learning and vision for road following
- Rahul Sukthankar, RACCOON ·aillight tracking, active vision
- Todd Williamson, YARF road following extensions, video compression

Other faculty: Martial Hebert, Takeo Kanade, Charles Thorpe
Other PhD students: Omead Amidi, Todd Jochem, Jennifer Kay, Karl Kluge, Ken Rosenblatt
Staff: Jay Gowdy (became graduate student in August 92), Bala Kumar, Dirk Langer, Jim Moody, Bill Ross

1.3 Publications

- Crisman, J., Thorpe, C. SCARF: A Color Vision System that Tracks Roads and Intersections. In *IEEE Trans. on Robotics and Automation*, in press.
- Hebert, M. (1992) 3-D Landmark Recognition from Range Images, Proceedings CVPR'92, Champaign, Illinois.
- Hebert, M., Krotkov, E. (1992), 3D Measurements for Imaging Laser Radars, in *Image and Vision Computing 10:3.*
- Kluge, K., (1992), Representation and Recovery of Road Geometry in YARF, in *Intelligent Vehicles 92*, I. Masaki ed.
- Langer, D., Thorpe, C. (1992) Sonar based Outdoor Vehicle Navigation and Collision Avoidance; *Proceedings IROS 1992*; July 7-10, Raleigh, NC.
- Pomerleau, D.A. (in press) Knowledge-based Training of Artificial Neural Networks for Autonomous Robot Driving. To appear in *Robot Learning*, J. Connell and S. Mahadevan (eds.), Kluwer Academic Publishing.
- Pomerleau, D.A. (1992) Progress in Neural Network-based Vision for Autonomous Robot Driving. In Proceedings of the 1992 Intelligent Vehicles Symposium.
- Pomerleau, D.A. (1992) Neural network perception for mobile robot guidance. PhD. Dissertation. Carnegie Mellon technical report CMU-CS-92-115.
- Pomerleau, D.A., Gowdy, J., Thorpe, C.E. (1991) Combining artificial neural networks and symbolic processing for autonomous robot guidance. In *Engineering Applications of Artificial Intelligence, 4:4* pp. 279-285. Also published in *Proceedings of the 1992 DARPA Image Understanding Workshop.*
- Pomerleau, D.A. (1991) Efficient Training of Artificial Neural Networks for Autonomous Navigation. In *Neural Computation 3:1* pp. 88-97.
- Reece, D. (1992) A Tactical Control System for Mobile Robot Driving, PhD Dissertation, Carnegie Mellon
- Thorpe C., Amidi O., Gowdy J., Hebert M., Pomerleau D. (1991) Integrating position measurement and image understanding for autonomous vehicle navigation. In *Proceedings of the Second International Workshop on High Precision Navigation*, Stuttgart, Germany.

Chapter 2

Sonar-Based Outdoor Vehicle Navigation and Obstacle Avoidance

2.1 Introduction

The autonomous land vehicle Navlab has already successfully been driven on roads and cross country. Different sensors are used to perceive the structure of the environment and navigate the vehicle under a variety of conditions as described for example in [8]. The sensors mainly employed so far were color video cameras and the ERIM 3-D laser range finder. Detecting obstacles and taking an appropriate decision is an important task for any mobile system in order to navigate safely through its environment. Obstacle detection is possible using colour video images or ERIM range images. However, owing to the data aquisition process and the required intensive image processing, these types of perception systems are generally not very well suited for a quick reaction to unexpected obstacles. Especially in the case of collision avoidance, a sensor is needed that can supply the relevant information fast with little data processing overhead and interact with actuators at the level of the vehicle controller (See also [1]). Sensors that satisfy these requirements are for example sonars, infrared sensors, pulsed 1-D laser range finders or microwave radar[1].

Compared to light based sensors, sonars have the advantage that they do not get confused by transparent or black surfaces. On the other hand, the wavelength of ultrasound is much larger than the wavelength of light, i.e. usually around 4 mm as compared to 550 nm for visible light. Therefore, unless the transducer faces the reflector surface in a normal direction, only rough surfaces or edges can reflect sound waves. However, most real world outdoor surfaces almost always have a type of surface roughness that enables a sonar to detect the object. It was therefore decided to use sonar sensors for the collision avoidance system of the autonomous land vehicle Navlab. The following sections describe the sonar system and its performance in an outdoor environment. Some novel results were obtained in using the system for vehicle navigation by itself and by integrating it into other vehicle navigation systems. The system is configured in such a way that more sensors can be added easily in the future. These sensors do not necessarily have to be sonars but can be any other type of point range sensor. In the future it is intended to integrate at least one other type of range sensor into the system, most probably laser or radar based.

[1] An earlier version of this paper appeared as Sonar based Outdoor Vehicle Navigation and Collision Avoidance, by Langer, D., Thorpe, C., in *Proceedings IROS 1992*

2.2 Hardware Configuration

Sonar sensors have already been used successfully for indoor mobile robots as described in [2] and [4]. An outdoor environment, however, adds some additional constraints on the type of sensor that can be used. Specifically the sensor should be robust against moisture, dust particles and noise from the vehicle engine and other sound sources.

Therefore an open type electrostatic transducer such as the Polaroid cannot be used. Instead a closed type piezoceramic transducer operating at a frequency of 80 kHz was selected. A detailed description of this sensor is given in [5]. The high operating frequency makes this sensor fairly robust against acoustic noise, while still providing an operating range up to 6 m. The beam angle of the sensor is approximately 5°, i.e. at the 3 dB intensity fall off from the major axis. Based on these characteristics, a total of five sensors was chosen in order to provide a good area coverage in front of the vehicle with a reasonable spatial resolution. The sensors are mounted on a guide rail such that their position and orientation can be freely adjusted. A typical sensor arrangement is shown in Fig. 2.1. Certain sensor configurations or environments can lead to acoustic interference between individual sensors. Therefore the hardware provides the ability to choose the exact trigger time of each sensor. In most circumstances the sensors are mounted such that they point away from each other. In this case all sensors are triggered at the same point in time. At present a measurement rate of 9 Hz is used, which is based on the following calculations: For very good reflectors we can assume a maximum operating range of 8 m, which corresponds to a time of flight of sound in air of approximately 50 ms. Thus echoes are considered during a receiving period of T_{rec} = 50 ms after triggering the sensor. In order to avoid measurement errors due to multiple echoes, only the range of the first echo is measured. The sensors are retriggered after an additional wait period of T_{wait} = 60 ms, which ensures that all detectable echoes from previous pulses are attenuated below the detection threshold. Thus the total cycle time $T = T_{rec} + T_{wait}$ = 110 ms. Each sensor measurement is tagged with the position of the vehicle. At present the Navlab uses dead reckoning to estimate its position relative to some initial point. The distance travelled is provided by an optical encoder on the drive shaft and vehicle orientation in 3-D space is provided by the gyroscope of an inertial navigation system. The measurements are combined to give the x-y position and orientation of the vehicle with respect to world coordinates. For the position tag of a sonar measurement only the following three parameters are used: x, y and φ, where φ = yaw (Refer to Fig. 2.1).

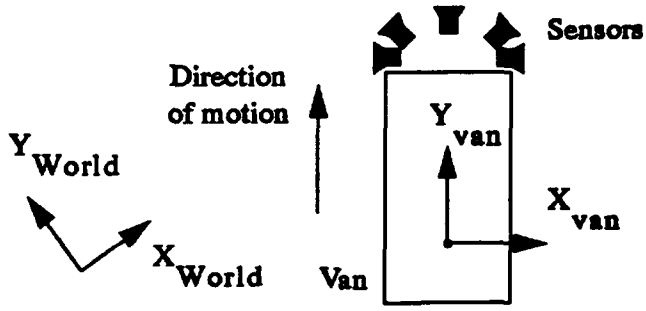


Figure 2.1: Sensor Configuration

The hardware of the sonar system consists of an interface module which triggers the sensors and measures the time of flight to the first echo returned. The interface module is accessed via VME bus from a 68020 based CPU. This processor runs as a slave of the vehicle controller processor in a real time environment and takes care of data aquisition, conversion to range, position tagging and proper timings. The map building and tracking algorithms are presently implemented on a Sun SPARC station which communicates with the vehicle controller via ethernet. Ethernet communication time uncertainties can be neglected because of the comparatively long cycle time of the sonar system.

2.3 Sonar Map

The previously described sensor system can now be used to build a local grid map. The grid map is called local because it contains only information about the immediate surrounding of the vehicle. The vehicle position is kept at a fixed point in the map. As the vehicle moves, objects in the map are moved from cell to cell relative to vehicle position. Once an object falls outside the map boundary it is discarded and the information is lost. Using just a local map has the advantage that error accumulation owing to dead reckoning is kept small, since only relative movements are considered. On the other hand the disadvantage is that information is lost and thus no global information is available. However, if desired, sequences of the output from the local map could be combined and included in a larger global map. At present the area covered by the local map is 16.4 m x 16.4 m. Each grid cell is taken to cover an area of 0.4 x 0.4 m^2. Hence the map consists of 41 x 41 cells (See Fig. 2.2). .

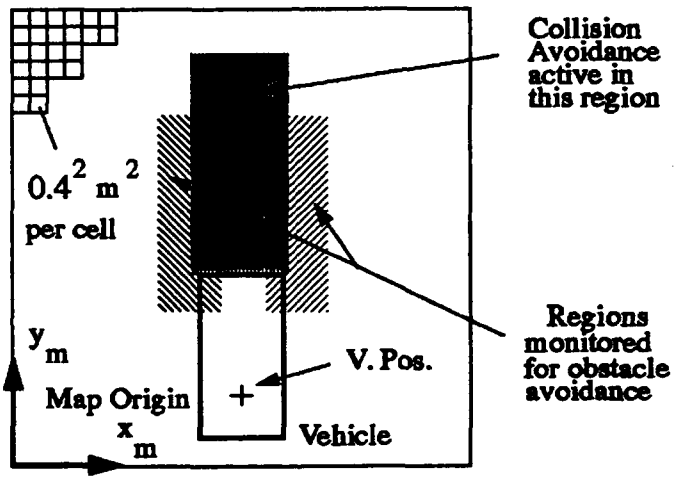


Figure 2.2: Sonar Map

The reason for taking such a coarse resolution for each cell was that most applications of the system do not require a high accuracy. Also, the sensor itself does not always provide a highly accurate measurement. For the particular sensor chosen, the measurement accuracy is about 1 cm. However, depending on the environment, the sensor may also deliver noisy results. The reason lies in the poor angular resolution of the sensor. Echoes may be detected from an object at far range in the main lobe or from a good reflector at closer range in a side lobe. Depending on the relative signal strenghts and movement of the sensor, the range reading may oscillate. A similar effect can also happen when an echo reflected multiple times is received.

Each cell has a set of parameters or annotations associated with it, which are described below:

1. **Object Type**: This parameter is used to indicate if the object in that cell was seen at the current sensor reading, or if it was seen at a previous reading. If it was seen only at a pre- vious reading, then the object type indicates that it must have moved to that particular cell due to vehicle motion only.
2. **Position**: Indicates the x-y position of the object with respect to vehicle position.
3. **Count**: This parameter counts the number of times an object was detected in a particular cell.

The resolution of the grid is fairly coarse and hence a position parameter (X_{obj}, Y_{obj}) is kept to avoid gross error accumulation when objects are transformed in the map. Only one object is kept per grid cell. Thus measurement uncertainty is part of the grid cell representation and any object detected within an area covered by a particular cell is taken to belong to the same object.

Following, a short description of object transformation within the map is given: Vehicle position and orientation are kept constant within the map. Therefore objects in the map move with respect to the vehicle. The vehicle's positioning system returns vehicle position and orientation with respect to a global frame of reference (x, y) that is determined at the time of initialization of the system. Since we are interested only in the movement of objects with respect to the vehicle's coordinate system (x_m, y_m), the appropriate transformation is obtained by using the increment in travelled distance and orientation . Depending on the type of positioning system of the vehicle, errors due to dead reckoning are reduced and the positioning system can be initialized independently by using this method. Hence if the vehicle moves from a position in a plane (x_1, y_1, φ_1) to a new position (x_2, y_2, φ_2), then an object in the map at position (x_{m1}, y_{m1}) is transformed to position (x_{m2}, y_{m2}) as follows (Fig. 2.3):

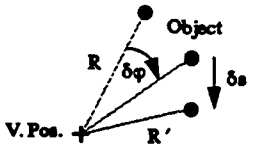


Figure 2.3: Object Transformation in Sonar Map

Position and orientation increment:

$$\delta s = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad \delta\varphi = \varphi_2 - \varphi_1 \tag{2.1}$$

Transformation parameters:

$$R = \sqrt{x_{m1}^2 + y_{m1}^2} \quad \alpha = \arctan\frac{y_{m1}}{x_{m1}} \quad \alpha_T = \alpha - \delta\varphi \tag{2.2}$$

New object position (vehicle moving forward):

$$x_{m2} = R\cos\alpha_T \quad y_{m2} = R\sin\alpha_T - \delta s \tag{2.3}$$

After the position of objects in the map is updated, new objects detected by the sensors are added. A sensor measures the range R to an object. Position and orientation of each sensor on the vehicle are known. Hence, using the transformation 'vehicle position to sensor position to map', a new object is placed in a cell in the map (Refer to Fig. 2.4). If that particular cell is already occupied, then only the cell parameter **Count** is updated, otherwise all cell parameters are updated.

The map parameter **Count** is used for differentiating between moving and stationary objects and for filtering the data. **Count** can also be used to evaluate the confidence that a particular cell is occupied by an object. A higher value of **Count** indicates a higher confidence. In the case of collision avoidance for example it is desirable to slow down the vehicle if there is an obstacle in front and to resume driving if the obstacle moves away, like a car or person could. Hence objects in the map in a sensor's field of view are deleted if a sensor does not detect them anymore. However an object will not be deleted instantly, but only after it was not seen by the sensor for a certain time period. This time period is defined by a parameter called **Life Time**, which is given as the number of cycles of the update sequence

(Fig. 2.6). We can assume that the cycle period is approximately constant. The parameter **Count** is therefore updated as follows:

If an object is detected, then

1. Initially, $Count_t$ = Life Time
2. Otherwise, $Count_t = Count_{t-1} + 1$
3. Also, Decay Amplitude = 0

If no object is detected by the sensor and **Count** is not equal to zero, then

1. Calculate Decay Amplitude initially, i.e. if it is zero: Decay Amplitude = $Count_t$ / Life Time

   Decay Amplitude is calculated only once at the begin- ning of a decay sequence. It ensures that each object disappears after the same amount of time, i.e. **Life Time**, has passed.
2. $Count_t = Count_{t-1}$ - Decay Amplitude

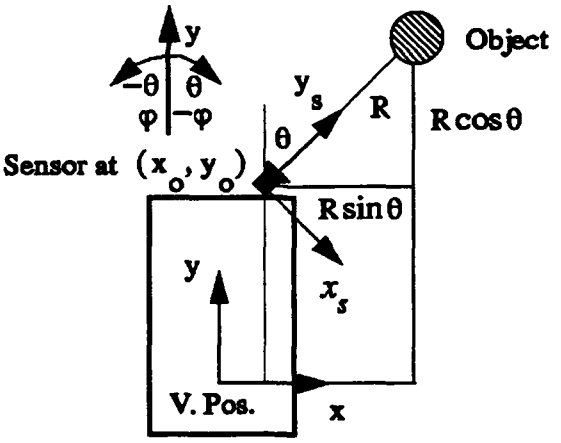


Figure 2.4: Sensor to Object Transformation

At present, the decay algorithm is applied only to objects appearing in the area directly in front of the vehicle. Moving objects appearing in this area are of most concern for safe vehicle navigation. Therefore the velocity of the vehicle is reduced as a function of range to the closest object detected in this area. The velocity is set to zero if the closest range is less than a certain minimum distance. The decay algorithm ensures that the vehicle resumes driving when the obstacle moves away. Fig. 2.5 shows a plot of the percentage of vehicle velocity set versus closest range. D_{max} corresponds to the maximum sensor operating range.

The parameter **Count** can also be used to eliminate spurious echoes such as the ones returned by rain droplets. In this case an object is supposed to be actually present only, if it has been seen consecutively for a certain number of

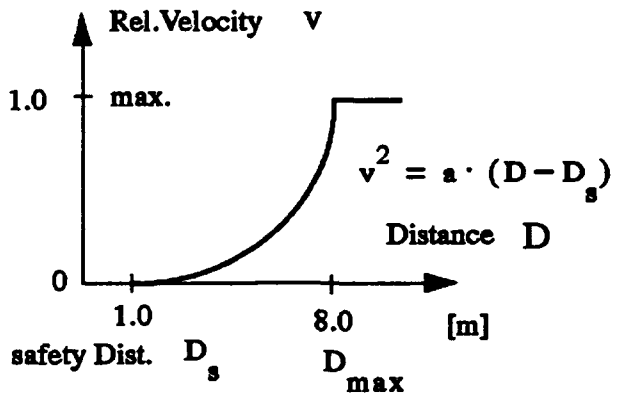


Figure 2.5: Velocity control

cycle times. Since rain droplets return echoes at random ranges as time progresses, these returns can be filtered out and will not appear as ghost objects in the map.

Of course this procedure does not work in a heavy downpour since the number of rain drops in the environment just becomes too large.

The following figure shows the cycle for map operations:

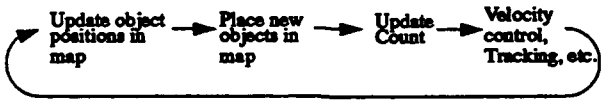


Figure 2.6: Updating sequence for sonar map

The parameter **Object Type** is used to indicate the type of operation that was performed on an object in a particular grid cell. These operations can be transformations as mentioned before, or filtering operations like the ones decribed in the next section.

At present, each grid cell is associated with only three parameters. The map structure allows an extention to other annotations (pointers to data structures) in the future, such as object characteristics or triggers. In this way the sonar map could easily be integrated into an *Annotated Map* such as the one described in [7].

2.4 Feature Selection and Tracking

The data collected by the sonar map can now be used for autonomous vehicle navigation functions. A basic navigation function is the tracking of features in the environment and using this information to determine a path that the vehicle can drive. The following paragraphs describe a method by which the vehicle uses its sonar sensors to drive on a path parallel to a feature such as a wall, railroad track or parked cars.

Fig. 2.9 shows data collected in the sonar map when tracking cars parked on the right hand side of the road.

As can be seen in the figure, the side of the cars facing the road is fairly well detected. Usually sonar does not detect smooth surfaces very well because of specular reflections. However, in most real world environments such as this one, there are no perfectly smooth surfaces. In this case the sonar receives echo returns from corners and projections like door handles, mirrors, wheels, etc..

The vehicle is to be driven on a path parallel to the curve formed by the parked cars, keeping a constant distance from the cars (usually around 3 m). Therefore the parameters of that curve have to be calculated first, using the data from the sonar map. For reasons of computational simplicity and decreased noise sensitivity a least square fit of a straight line was chosen. All computations are performed with respect to the vehicle origin which is at a fixed position in the sonar map. Data points for the line fit are selected by choosing only data points that appear in a specific area in the sonar map. Thus for the environment represented in Fig. 2.9, only the right half of the map is searched for data points. The Position parameter gives a data point in the map in terms of vehicle coordinates. Since the direction of vehicle motion is along the y-axis, a line parallel to the vehicle would have a slope of $m = \infty$ (Fig. 2.2; Note that this coordinate system is defined by the vehicle's position system). To avoid this inconvenience, the vehicle coordinate system is rotated counterclockwise by 90. All following computations are now performed with the transformed coordinates $y = -x_{old}$ and $x = y_{old}$. Therefore the selected data points can now be represented as a discrete function $y_i = f(x_i)$. The sonar sensors sometimes return a spurious echo. These outliers generally degrade the performance of a least square fit. Hence a median filter is applied first on the data points given by $y_i = f(x_i)$. The filter is applied twice, using window sizes three and five cells. The two parameters of the straight line $y = mx + c$ can now be found by using standard formulae for a least square fit for n data points (x_i, y_i):

$$m = \frac{n\sum x_i y_i - (\sum x_i)(\sum y_i)}{n\sum x_i^2 - (\sum x_i)^2} \quad c = \frac{\sum y_i}{n} - m\frac{\sum x_i}{n} \tag{2.4}$$

Also, the standard error of estimate is given by:

$$s_o = \sqrt{\frac{\sum (y_i - (mx_i + c))^2}{n - 2}} \tag{2.5}$$

The data trajectory parameters and are stored and updated during each system cycle (Refer to Fig. 2.6. The line can now be used by a path tracker to steer the vehicle [1]. To ensure a consistent steering response, the line fit also has to be checked for validity. It may happen that the sensors do not detect any features for some driving distance. One reason in the case of parked cars for example may be a gap between cars and no reflectors on the road edge. Here the line fit will produce no valid output. The standard error s_o is used to provide a measure of how well the data points could be fitted. If s_o is too high then again the output of the line fit is not taken to be valid. In these cases the old path parameters are used and the vehicle will continue driving in the previously calculated direction until it encounters features again. Since the old path parameters are referenced to the vehicle position they still have to be updated to compensate for vehicle movement during one system cycle. The position and orientation increment during one system cycle is known from . Hence m is transformed as follows:

$$m_{new} = \tan(\varphi - \delta\varphi) \quad \varphi = \arctan(m) \tag{2.6}$$

In order to obtain c_{new}, point $P(x_1, y_1)$ is selected on the path where it intersects the y-axis (for convenience) as shown in Fig. 2.7. Using 2.2 and 2.3, the coordinates of P in the new coordinate system (x_{1new}, y_{1new}) can be calculated. Hence c_{new} can be calculated by

$$c_{new} = y_{1new} - m_{new}x_{1new} \tag{2.7}$$

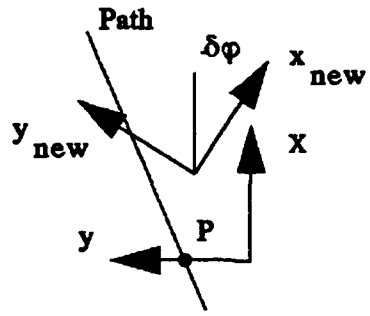


Figure 2.7: Path parameter transformation

The method described above worked well in an environment that provided good reflectors and continous smooth features. However, especially in the case of parked cars, problems arise when gaps are encountered or reflectors that do not belong to the feature being tracked are near by. A typical situation is shown in Fig. 2.8. In the least square fit sequence 1 - 4 , only line fits 1 and 4 track the feature. Type A data does not belong to the feature being tracked, type B data is due to corner effects at a gap and type C data *is a noisy range measurement.* Type B data can lead to an undesired least square fit as shown by line fits 2 and 3. In order to reduce these errors, obtain a smoother steering response and make the system more robust against outliers, the values obtained from the least square fit are filtered and path parameters are updated by merging new information with past values. The method is described in the following paragraphs:

An initial noise reduction is achieved by selecting data points only from a small window in the sonar map: As we have some knowledge about the environment the vehicle is driving in, we can predict up to a certain degree where we should look for valid data in the map. This means in practice that only data points within a certain distance from the data trajectory are taken. This procedure removes outliers of type A as shown in Fig. 2.8. Furthermore a point is selected only if it is within a certain distance and direction from previously selected adjacent points. This method removes outliers of type C and ensures that most points are already grouped close to a line segment.

Again a straight line is fitted using 2.4. For each distance travelled, we obtain a new value of gradient m_i. The change in gradient is then given by

$$\delta m_i = m_i - m_{i-1} \tag{2.8}$$

and $\delta\varphi_i$ is the corresponding change in vehicle orientation during that interval. Since not all type B data is removed, there still may remain a problem with sudden large changes in orientation as shown by the least square fit sequence 1 - 4 in Fig. 2.8.

These sudden changes in orientation basically appear as median noise. They can be filtered out by putting current and past readings of δm into a buffer of N values, passing a median filter across and then averaging over N values. The buffer is implemented as an ordered set,

$$m_{Buffer} = \{\delta m_i, \delta m_{i-1}, \delta m_{i-2}, \ldots, \delta m_N\} \tag{2.9}$$

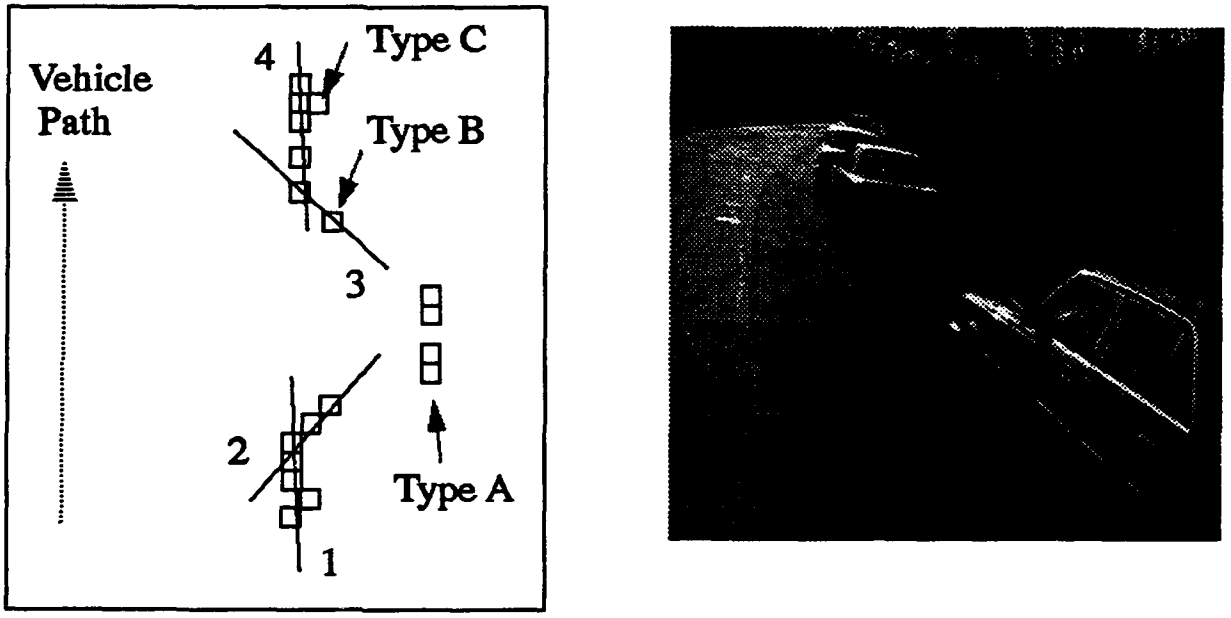


Figure 2.8: Removing outliers from least square fits: (a) Map display, (b) Typical corresponding scene

During each update sequence, all elements are shifted right by one, the last element being discarded and the current result replacing the first element. Median filtering is achieved by replacing element δm_{i-k} with a median value, where $2k+1$ is the maximum possible window size of the filter. Since values towards the right in the buffer represent increasingly earlier points in time/ distance, successive applications of a median filter can be achieved by replacing elements δm_{i-k-t} with the filtered value, where t is the discrete shift in time/distance. The new value m^{new}_{Path} of the path parameter is then computed, compensating also for change in vehicle orientation during the averaging interval,

$$m^{new}_{Path} = \frac{1}{N} \cdot \sum_{k=1}^{N} \tan\left(\arctan(\delta m_k) - \sum_{i=1}^{k} \delta\varphi_i\right) \tag{2.10}$$

m_{Path} is then updated by merging the current and the new value for m_{Path} using a weighted average,

$$m^{t+1}_{Path} = \frac{m^{new}_{Path} + w \cdot m^{t}_{Path}}{1+w} \quad w \geq 0 \tag{2.11}$$

In a similar fashion as described above, the path parameter c_{Path} is updated. The weight w and the number of values N control how close the path parameters m_{Path} and c_{Path} should follow the actual data points. If a lot of noise is present in the data, the path parameters should be influenced only slightly, whereas if little noise is present, the data should be followed closely. An estimate of the noise present is given by the standard error s_o from 2.5 and w and N are adjusted accordingly. As a result the steering of the vehicle is now much smoother.

A drawback of the method is its slow reaction to a relatively sudden change in road direction. This effect is also due to the short range of the ultrasonic sensors and the fact that almost no echoes are received at large angles of incidence. In the case of features being tracked on the right side, the system is able to handle curves to the right since data points slowly move away from the vehicle and the vehicle can follow with a slight delay. On the other hand a curve to the left poses a problem because by the time the vehicle recognizes that it should change direction, it has

usually come already too close to the feature and has no space left to make a sharp left turn anymore. For this reason a monitor is added which monitors a particular area for objects towards the right hand side of the vehicle as indicated in Fig. 2.2. If objects are detected in this area, a steering radius R_m necessary to avoid the closest object is calculated. The value of R_m is given by a relation between R_m and the closest distance D_c similar to the one for velocity control as shown in Fig. 2.5. Velocity is here replaced by steering radius and for example for D_c between 0 and 2 m, varies between -20 and -100 m. 28 D_c is defined as the closest distance between the objects and a point at the front right edge of the vehicle. If R is now the steering radius calculated by a path tracker as described in [1] and if $R > 0$ or $R < R_m$, then R_m overrides R and the vehicle follows steering commands issued by the monitor. Otherwise the path tracker takes over again. Similarly a monitor can be used when tracking features on the left side. Note that steering radii are negative when turning left and positive when turning right.

In general the system performs well when tracking a wall or a feature that changes curvature smoothly. When tracking parked cars it does not perform as well in curves and fails when the road curvature becomes sharp. The reason here is that often parked cars are not very well aligned and even on straight road stretches parked at different angles to each other. The maximum range of the sensors is simply too short to detect these configurations well enough. Fig. 2.9 shows a preliminary result of the vehicle tracking parked cars, detecting a gap and preparing for reverse parking.

2.5 Results and Conclusions

The sonar system successfully drove the Navlab on a dirt road next to a railroad track, using the railroad to guide the vehicle. Parking or docking maneuvres are also important autonomous vehicle tasks. At present the sonar system is used to drive the vehicle parallel to parked cars in a city street. Eventually the system should be able to detect a parking space and autonomously park the vehicle. The sonar is also successfully integrated into two other systems that drive the Navlab. The first one is YARF [3], which drives the robot on city streets. YARF uses colour vision for road following but cannot detect if obstacles obstruct the vehicle's path and thus cannot adjust the velocity accordingly. The sonar system takes over that task and sends velocity commands to YARF via the EDDIE toolkit [7]. The sonar is also integrated into the new architecture DARN (Distributed Architecture for Robot Navigation) of the Navlab. The outputs of several modules that can drive the vehicle are used here to decide on an optimum path for the vehicle [6]. The sonar module sends all object positions with respect to vehicle position from the sonar map and the system selects an obstacle free path. Communication is again facilitated via the EDDIE toolkit.

The sonar system proved to work reliably in a variety of different situations. There were no major problems with false returns or sensor noise that could not be dealt with. One reason is most probably that an outdoor environment like a road is generally less cluttered than most indoor environments where autonomous vehicles are used. Outdoor objects tend to be large, having usually enough corners and projections that reflect ultrasound well. Care has to be taken in avoiding reflections from the ground. This problem can be solved in most cases by mounting the sensor high enough above the ground and pointing it slightly upward. The system can also be easily integrated with other vehicle navigation systems or adapted to other vehicles.

A drawback of using ultrasound in air is the limitation of range and data update due to high attenuation and low speed of sound. At present this fact does not matter that much since the system is used only at low vehicle speeds. However, the current system design is not limited to using only ultrasonic sensors. A microwave radar, laser scanner or stereo camera can be used as well. In the future therefore we intend to integrate some of these sensors.

2.6 References

[1] O. Amidi. Integrated Mobile Robot Control. Technical report, Robotics Institute, Carnegie Mellon University,

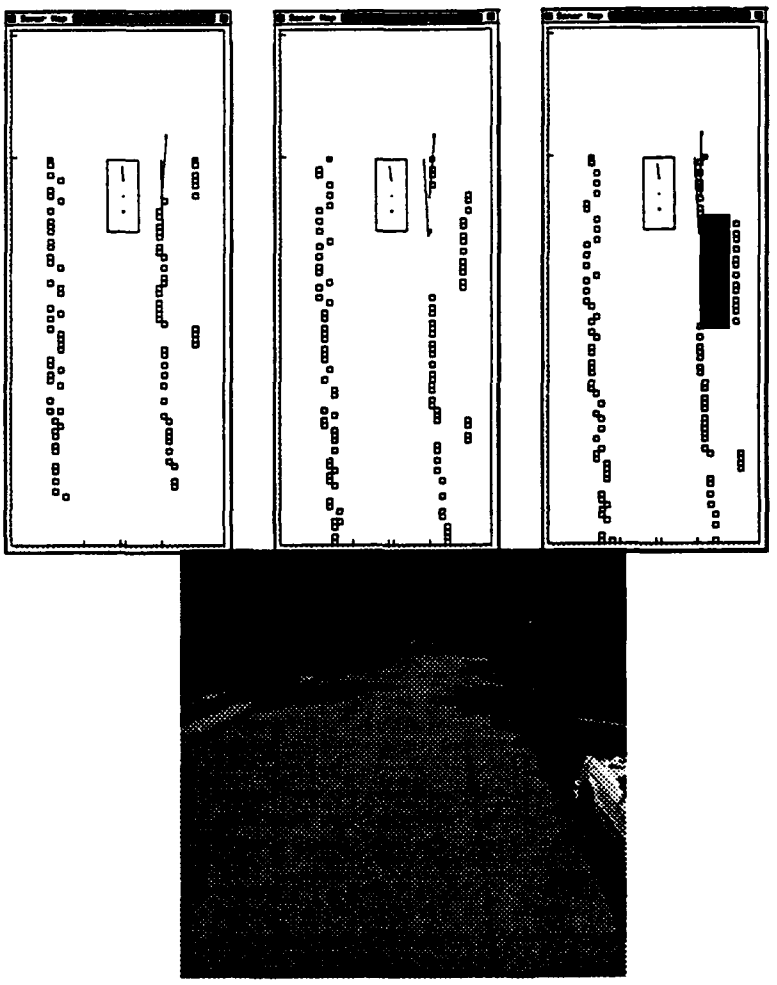


Figure 2.9: Tracking parked cars on the right hand side and searching for parking space: (a) Approaching gap, (b) Detecting gap, (c) Preparing vehicle for parking maneuvre, (d) Typical street scene.

1990.

[2] A. Elfes. A Sonar-Based Mapping and Navigation System. In *Proc. IEEE Conference on Robotics and Automation*, 1986.

[3] Karl Kluge and Charles Thorpe. Explicit Models for Robot Road Following. In *Vision and Navigation: The Carnegie Mellon Navlab*. Kluwer Academic Publishers, 1990.

[4] J. Leonard and H. Durrant-Whyte. Application of Multi-Target Tracking to Sonar-based Mobile Robot Navigation. In *Proc. IEEE Conference on Decision and Control*, 1990.

[5] V. Magori and H. Walker. Ultrasonic Presence Sensors with Wide Range and High Local Resolution. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, UFFC-34(2), 1987.

[6] David W. Payton, Kenneth Rosenblatt, and David M. Keirsey. Plan Guided Reaction. *IEEE Journal on Systems, Man and Cybernetics*, December 1990.

[7] C. Thorpe and J. Gowdy. Annotated Maps for Autonomous Land Vehicles. In *Proceedings of DARPA Image Understanding Workshop*, September 1990.

[8] Charles Thorpe, Martial Hebert, Takeo Kanade, and Steven Shafer. Toward Autonomous Driving: The CMU Navlab. Part I - Perception. *IEEE Expert*, August 1991.

Chapter 3

Evaluation of 3-D Measurements From Imaging Laser Radars

3.1 Introduction

Range sensing is a crucial component of any autonomous system. It is the only way to provide the system with three-dimensional representations of its environment. The classical computer vision approach to range sensing is to use passive techniques such as stereovision, or shape from X. However, those techniques are not yet sufficiently reliable or fast to be used in many applications, most notably real-time robotic systems. Active sensors, which generate the illumination instead of using only the ambient illumination, have received increasing attention as a viable alternative to passive sensors. Their importance was recognized relatively early. For example, Nitzan et al. [13] describe a system for interpreting indoor scenes that uses range and intensity from a laser ranging system. Many such sensors were developed [1] and many have been used in real computer vision and robotics applications such as obstacle avoidance [2, 17] and autonomous navigation [10] of mobile robots. Surveys of rangefinding sensors and their applications in robotics can be found in [4, 8, 12]. A review of their use in autonomous navigation of mobile robots can be found in [6] [1]

Although significant theoretical results have been derived in the area of laser radar characterization [16], experimental work is needed to evaluate performance in robotics applications. In this paper, we characterize and evaluate a class of sensors, the imaging laser radars[2]. Those sensors have been proposed as a good compromise between accuracy, resolution, and speed requirements, especially in the context of mobile robotics. Our intent is to present measurement and noise models for those sensors, to identify problems that are specific to this class, and to provide experimental data to support our conclusions. Our emphasis is on identifying limitations and capabilities that have an impact on the use of standard image analysis algorithms for those sensors.

We concentrate on laser radars because of our experience with two such sensors, Erim and Perceptron, in our work in mobile robotics. The theoretical and experimental results presented in this paper use in part results from earlier analyses from [3, 9, 18, 20] for the Erim sensor, and from [11, 15] for the Perceptron sensor. However, we suggest that more analyses of this type are needed in order to better grasp the state of sensor technology from the point of view of computer vision and robotics.

[1] A different version of this chapter first appeared as 3D Measurements for Imaging Laser Radars, by Hebert, M., and Krotkov, E., in *Image and Vision Computing 10:3*

[2] An optical-wavelength radar is also called Lidar, which is an acronym for Light Detection And Ranging.

The paper is organized in three parts. Section 3.2 describes the principle of the sensors, the theoretical models of noise and measurement geometry, and the two sensors that we use in our experiments. Section 3.3 describes some problems that are specific to this class of sensors. Those problems can significantly impact the quality of the data and limit the use of those sensors. We distinguish between problems that are inherent to the physics of the laser radars, and problems specific to the hardware currently available for robotics applications. Section 3.4 describes experimental results obtained from actual sensors. The experiments illustrate the measurement models and problems introduced in the previous two sections.

3.2 Sensors

In this section we address imaging laser radars, covering both their principles and practical characteristics. First, we describe the general principle of operation, define a sensor reference frame, and present measurement models for the range and intensity data. Then, we describe two particular sensors that we used for experimentation.

3.2.1 Principle of Operation

The basic principle of a laser radar is to measure the time between transmitting a laser beam and receiving its reflection from a target surface. Three different techniques can be employed to measure the time of flight, which is proportional to the range: pulse detection, which measures the time of flight of discrete pulses; coherent detection, which measures the time of flight indirectly by measuring the beat frequency of a frequency-modulated continuous-wave (fm-cw) emitted beam and its reflection; direct detection, which measures the time of flight indirectly by measuring the shift in phase between an amplitude-modulated continuous-wave (am-cw) emitted beam and its reflection.

Experimental devices have been developed using both pulse and coherent detection technologies (for a survey, see Besl [1]). They are not yet widely in use in computer vision and robotics applications. In this paper, we concentrate on am-cw laser radars.

For am-cw laser radars, the range to a target is proportional to the difference of phase; if $\Delta\varphi$ is the difference of phase, then the range is $r = \frac{\lambda}{4\pi}\Delta\varphi$, where λ is the wavelength of the modulation. Since the phase is defined modulo 2π, the range is defined modulo r_a, where $r_a = \lambda/2$ is the distance (or *ambiguity interval*) corresponding to the maximum phase difference of 2π. Therefore, an inherent limitation of this principle of operation is that it cannot measure range uniquely, *i.e.*, it measures range only within an ambiguity interval. In practice, it is not possible to distinguish between ranges r and $r + r_a$ without employing external constraints. or two beams with different modulation frequencies.

We digress briefly to consider constraints imposed by autonomous navigation. First, the ambiguity interval should equal or exceed the maximum range of interest. Typically, this maximum range depends on vehicle velocity and the type of terrain. For example, for a maximum range of 20 m, $r_a \geq 20$ m and $\lambda \geq 40$ m. Second, safety considerations limit the power and wavelength of the laser diode used to generate the laser beam. The power of the diode is typically 100–300 mW, and the wavelength of the carrier signal is typically 700–1000 nm (near-infrared).

For many applications, *imaging* laser radars are essential. Imaging sensors generate a dense set of points structured as an image. Typically, image generation is achieved by two mechanically controlled mirrors that raster-scan the beam across a scene, measuring the range at regularly sampled points. Note that regular spacing in image space generally causes irregular spacing in the scene. For real-time applications such as autonomous navigation, the image acquisition time is limited. The combination of electro-mechanical parts and time constraints adds new complexity and new sources of errors that do not exist in non-imaging sensors, such as surface probes and surveying devices.

In addition to range, am-cw scanners measure the "strength" of the reflected beam, thus generating a second image that some call the *reflectance* image. To avoid confusion with surface reflectance, we will refer to it as the

intensity image. This image is similar to a TV camera intensity image, but is registered with the range image, and does not depend on the ambient illumination, Although relatively little work has been done using Lidar intensity images, we have found them to be useful for automatically determining the position and orientation of the sensor with respect to a walking robot [9], for tracking roads from a robot truck [6], and for object recognition [5].

3.2.2 Sensor Reference Frame

It is useful to convert the range pixels to points in space expressed with respect to a reference frame. In this section we define a standard reference frame attached to the scanner, and the relation between a pixel (row, column, range) and the coordinates of the corresponding point.

Figure 3.1 illustrates the reference frame. As shown, the y-axis coincides with the direction of travel of the laser beam projected through the central point of the scanner (i.e., the principal ray). The angle θ (azimuth) corresponds to a rotation about the z-axis. The angle φ (elevation) corresponds to a rotation about the x-axis.

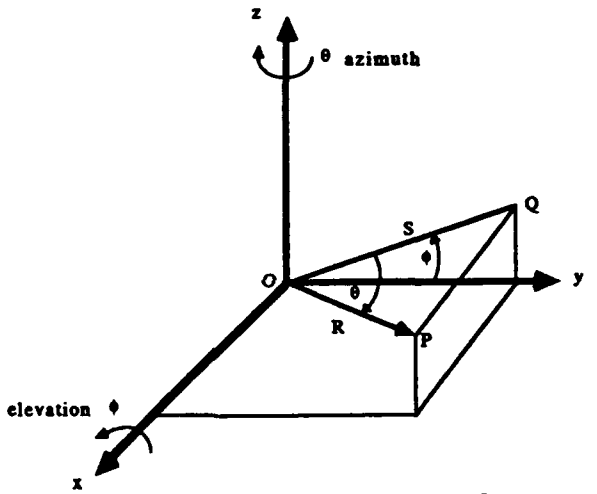


Figure 3.1: Reference frame for scanning laser range finder

In the figure, the symbol "R" denotes range. To be consistent with the majority of the text, we should denote range by "r."

Given the sensor measurement (u, v, d) (i.e., row, column, range), the transformation to spherical-polar coordinates is:

$$\varphi = u\Delta_\varphi + \varphi_0 \quad \theta = v\Delta_\theta + \theta_0 \quad r = \Delta_R d \tag{3.1}$$

where

- Δ_φ is the angular increment, in degrees/row, of the nodding mirror,
- Δ_θ is the angular increment, in degrees/column, of the panning mirror,
- φ_0 is the initial orientation, in degrees, of the nodding mirror,
- θ_0 is the initial orientation, in degrees, of the panning mirror, and

- Δ_R is the scanner range resolution in meters/grey-level.

Given the spherical polar coordinates φ, θ, r, the transformation to Cartesian coordinates is given by:

$$x = r\sin\theta \quad y = r\cos\theta\cos\varphi \quad z = r\cos\theta\sin\varphi \tag{3.2}$$

3.2.3 Measurement Models

An am-cw range sensor would approach perfection if it emitted a zero-width laser beam and observed the returned signal through an infinitely small receiver. In reality, the beam subtends a non-zero angle, and the receiver detects signals subtended by a solid angle we will call the instantaneous field of view (IFOV). Assuming a circular field stop, the beam projects to an ellipse on the target surface, the *footprint* of the beam. Every point within the intersection of the footprint and the IFOV contributes a range value and an intensity value to the final range and intensity measurements.

We may model a (range, intensity) pair as a complex number z, or equivalently as a vector (Figure 3.2a). The phase of z represents the sensed range (or phase shift), and the magnitude of z is the sensed intensity, According to this model, the range measured at a pixel is the integral of $k_0 z$ over the IFOV of the receiver, where k_0 is scalar and depends on parameters of the instrument (transmitted power, and the electro-optics of the receiver) and parameters of the environment (the angle α between the surface normal and the direction of measurement, the reflectance ρ of the target surface, and the range r as shown in Figure 3.2b).

Figure 3.2: Model of (range, intensity) pair as complex number

Ultimately, the fidelity of the range measurement depends on the power of the signal reaching the photodetector, which in turn depends on k_0. More precisely, the time-average radiant flux $\overline{F}_P$ (in watts) reaching the photodetector can be shown [16] to be:

$$\overline{F}_P = k_1 \frac{\rho\cos\alpha}{r^2} \tag{3.3}$$

where k_1 is a function of the transmitted radiant flux, the capture area of the receiver, and filter bandwidths. Assuming that the output power of the photodetector is proportional to $\overline{F}_P$, the output signal is proportional to $\rho\cos\alpha/r^2$.

When the received power is small, the output signal is small, and noise is significant. There are many sources of noise, including photon noise, laser noise, ambient noise, dark-current noise, secondary emission noise, and subsequent amplifier noise.

Nitzan et al. [13] identify photon noise as the dominant source, and assume that photoemission is a Poisson process to obtain the signal-to-noise ratio

$$\text{SNR} = k_2 \left(\frac{\rho\cos\alpha}{r^2}\right)^{\frac{1}{2}} \tag{3.4}$$

Assuming a modulation index of 100 percent, they go on to derive the approximate effect of photon noise on the measured range value. Their analysis indicates that the standard deviation of the range can be estimated by

$$\sigma_r \approx \frac{r_a}{\sqrt{2\pi}\,\text{SNR}} \tag{3.5}$$

Combining 3.4 and 3.5 yields

$$\sigma_r \approx \left(\frac{r_a}{\sqrt{2}\pi k_2}\right)\left(\frac{r^2}{\rho\cos\alpha}\right)^{\frac{1}{2}} \tag{3.6}$$

where the first term depends only on the physical, optical, and electronic characteristics of the sensor, and the second term depends only on the observed scene.

Our experimental results follow this model. In particular, Figure 3.3 shows the variation of σ_r^2 as a function of r, consistent with 3.6.

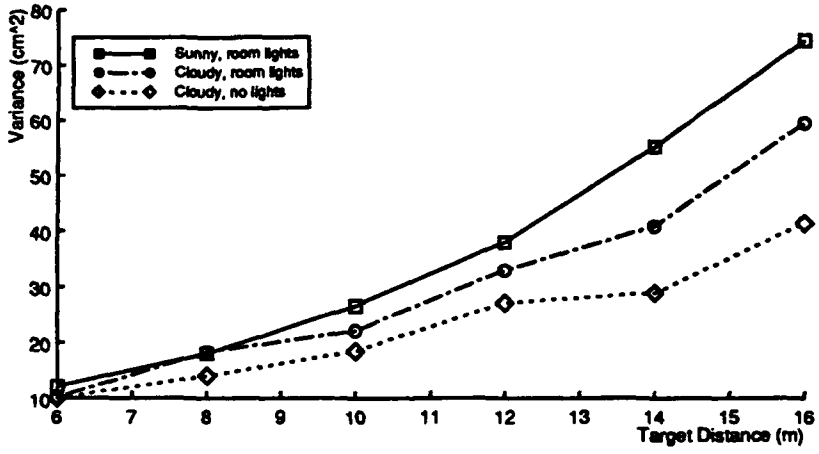


Figure 3.3: Range variance under different lighting conditions

3.2.4 Examples

We used two sensors for experiments: Erim and Perceptron, whose geometric parameters are listed below. Other sensors based on the same principle exist [1, 14]. The two sensors that we use are typical of the operation and performance of this type of sensor.

The Erim scanning laser rangefinder is designed for applications in outdoor autonomous navigation. Several versions of the scanner exist. We refer to the version used for research on autonomous land navigation [2, 6], which is the successor of a sensor used for legged locomotion [19].

The Erim scanner uses a 100 mW laser diode operating at 820 nm. The sensor volume is roughly 90 × 50 × 90 cm and the mass is about 45 kg. The scanning mechanism consists of a vertical rotating mirror for horizontal scanning, and a horizontal nodding mirror for vertical scanning. The acquisition rate is 0.5 s for a 64 × 256 × 8 bit image. Table 3.1 summarizes the characteristics of the sensor. Figure 3.4 shows a range image (top) and a reflectance image (bottom) from the Erim scanner. The images show an outdoor scene of a road surrounded by a few trees. The sharp discontinuity at the top of the range image is due to the ambiguity interval of about 20 m.

The Perceptron scanning laser range finder is more recent than the Erim, and has higher resolution and bandwidth. It is currently used for terrain mapping in support of legged locomotion [7].

The sensor volume is roughly 50×45×35 cm and the mass is about 30 kg. The sensor uses a 180 mW laser diode operating at 810 nm. The image acquisition rate is 0.5 s for a 256 × 256 × 12 bit image. The scanning mechanism is

Param.	Units	Erim	Perceptron	Description
φ_{fov}	deg	30	60	Vertical FOV
θ_{fov}	deg	80	60	Horizontal FOV
N_{rows}	pixel	64	256	Rows
N_{cols}	pixel	256	256	Columns
Δ_φ	deg	0.47	0.24	Vert. step (Note A)
Δ_θ	deg	0.31	0.24	Hor. step (Note B)
r_a	—	64 ft	40 m	Ambiguity interval
N	bit	8	12	Number of bits/pixel
Δ_R	—	3.0 in	0.98 cm	Range unit (Note C)

Table 3.1: Nominal values of sensor parameters

Note A: $\Delta_\varphi = \frac{\varphi_{fov}}{N_{rows}-1}$. B: $\Delta_\theta = \frac{\theta_{fov}}{N_{cols}-1}$. C: $\Delta_R = \frac{r_a}{2^{N_{range}}-1}$.

similar to the Erim design, except that the vertical field of view and the vertical image size are programmable. Table 3.1 summarizes the operating characteristics of the sensor as specified by the manufacturer [15].

Figure 3.5 shows the eight most significant bits of a typical range image (right) and intensity image (left) from the Perceptron. The distance between the scanner and the floor is approximately 4 m.

3.3 Problems

In this section, we describe four effects that may lead to corrupted or degraded data. Two effects, mixed pixels and range/intensity crosstalk, are due to fundamental limitations of am-cw laser radars, although they are sometimes compounded with problems in the design of the actual sensors that we used. The two other effects, distortion due to scanning and range drift, are more specific to the particular sensors that we use. However, it is important to be aware of those effects since they do affect the quality of the data. Furthermore, simple cost-effective remedies do not seem to exist at the moment even though those problems are theoretically avoidable.

3.3.1 Mixed Pixels

Significant problems occur at pixels that receive reflected energy from two surfaces separated by a large distance. From an image analysis point of view, we would like the range at such points to be measured on either of the surfaces, or at least to fall in between in some predictable way. The fact that the range is measured by integrating over the entire projected spot leads to a phenomenon known as *mixed pixels* in which the measured range can be anywhere along the line of sight. In practical terms, this means that occluding edges of scene objects are unreliable, and that phantom objects may appear due to mixed measurements that are far from the real surfaces. This is a problem inherent to direct detection am-cw laser radars and it cannot be completely eliminated.

Figure 3.6 shows the geometry of the measurement at an occluding edge: two objects at distances D_1 and D_2 from the sensor ($D_1 < D_2$) are separated by a distance D and a point is measured at the edge between the two surfaces. Due to the angular width of the beam, the range is formed by integration over a spot that contains reflections from both surfaces.The relevant parameter is the ratio p between the spot surface due to the near surface and the total area. The

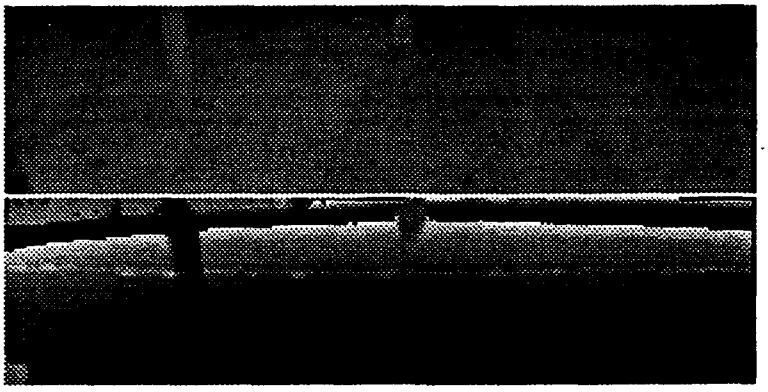


Figure 3.4: Erim intensity (top) and range images

The scene contains a tree (visible to the left), and a person (visible in the upper center) on a path.

combination of the two measurements is better explained using a model in the complex plane like the one presented in Section 3.2.3. Each portion i of the spot generates a measurement that can be represented by a complex number z_i. The phase of z_i, φ_i is proportional to the range D_i and its magnitude depends on the reflectivity ρ_i of the material. The resulting measurement is given by the sum $z = z_1 + z_2$. The final measured range is proportional to the phase of z.

The phenomenon becomes clear with this formulation: the phase of z can be anywhere between φ_1 and φ_2 depending on the ratio of the lengths of z_1 and z_2 which depends on p, ρ_1, and ρ_2. The effect of mixing on sensed range becomes even more unpredictable when D is greater than half the ambiguity interval of the sensor. In that case, the angle φ between z_1 and z_2 is greater than π and the resulting measurement is such that $\varphi_2 < \varphi < 2\pi$ or $0 < \varphi < \varphi_1$ depending on p and the reflectivities. In the first case, the range may be anywhere *behind* the furthest object. In the second case, the range may be anywhere *in front of* the nearest object. In practice, the latter situation occurs only in specific combinations of range and scene geometry. In practice, this phenomenon is observed as soon as there are discrete objects in the scene. Figure 3.8 shows the effect of mixed pixels in a real image. The top panel shows an Erim image with a small window (shown as a white rectangle) containing an object (a tree) and its left and right edges. The bottom panel shows an overhead view of the 3-D points in this region as calculated by Eq. 3.2, and using the range from the image pixels. The points at the center of the distribution correspond to the smooth surface of the object. Away from the center are two lines of mixed pixels that appear at the object's edges. In this example, all mixed pixels are located between the two targets, tree and background surface. This result is typical of the mixing effect in laser radars.

The mixed pixel effect complicates the processing and interpretation of the range images. Its main consequence is that strong range edges are unreliable. One approach to the problem is to apply a median filter to the image. The filter removes most of the mixed pixels since they appear as spurious readings only along the direction of an edge. Another approach is to transform all the points to a 3-D coordinate frame. There, mixed pixels appear as isolated points, which makes them easier to remove than in image space. Since these two approaches remove mixed pixels whose range is far from the true value, they will not remove those in the vicinity of the actual edge. Thus, it is not possible for them to remove reliably all the mixed pixels from the data.

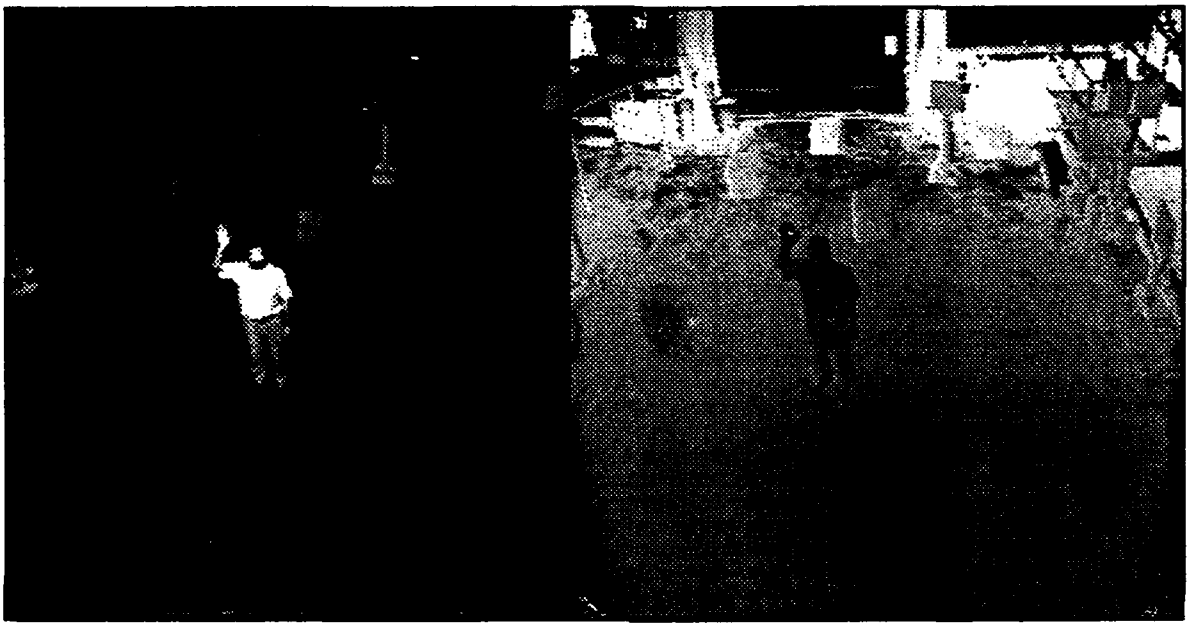


Figure 3.5: Perceptron intensity (left) and range images

A person is visible in the intensity image standing in front of box-like targets used for calibrating the sensor. The images have been enhanced for printing.

3.3.2 Scanning Pattern

With currently available commercial technology, imaging can be achieved only by scanning the beam using rotating and nodding mirrors. The scanning mechanism introduces additional errors into the sensor. They are probably the hardest to quantify and to correct. We identify three sources of error: synchronization, distortion, and localization. They result in a correct range measurement being stored at the wrong pixel in the image. Limitations due to the scanning mechanism are not inherent to the am-cw technology, but are due to the lack of alternatives to electro-mechanical scanning devices.

Synchonization The main problem, *synchronization*, is due to the fact that three systems (horizontal mirrors, vertical mirrors, and range measuring system) must be synchronized exactly. In particular, the motion of the two mirrors must be exactly synchronized with the sampling of the range measurements. A small error in synchronization results in an error in either φ or θ depending on which mirror is affected. Even though the range measurements themselves are not affected, the angular errors will propagate to the coordinates computed from Eq. (2). For example, poor synchronization may occur at the top of the image because the nodding mirror takes a finite amount of time to go from

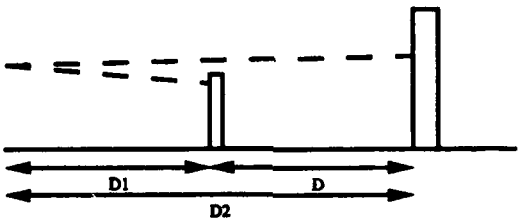


Figure 3.6: Surface 1 occludes surface 2 (side view)

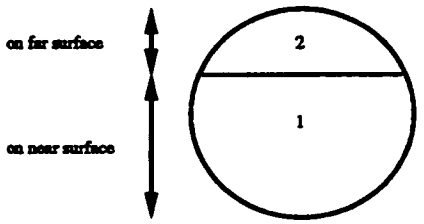


Figure 3.7: Mixed spot contains reflections from two surfaces (view from sensor)

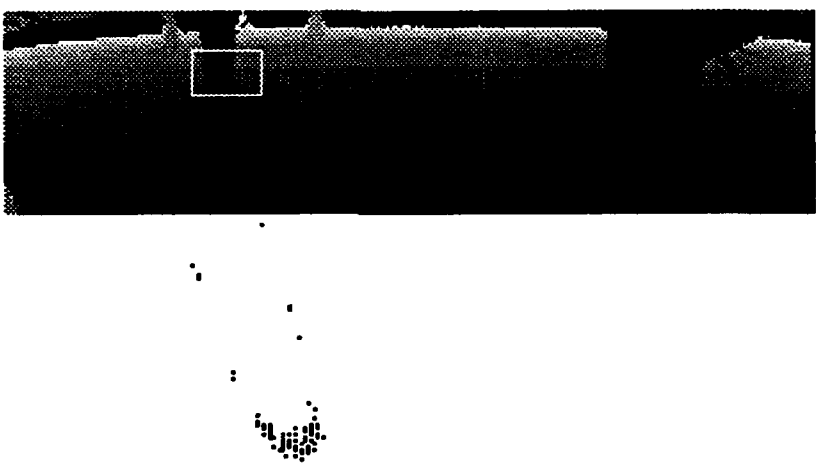


Figure 3.8: Mixed pixels in real image

Top: selected region in Erim image. Bottom: overhead view of the corresponding points

zero speed at its starting position to its normal scanning speed. During this interval of time, a few scanlines that are not correctly sampled are collected. Figure 3.9 shows a Perceptron image in which a rectangle in the scene projects to a skewed shape instead of a rectangle. This is due to the poor synchronization of the mirrors at the beginning of the scan. In this particular case, the problem can be fixed easily by starting the data acquisition a few lines below the starting position of the nodding mirror. In general, it is not possible to perfectly synchronize the mirrors. In general, there is a discrepancy between the nominal values of the scanning angles and the actual values. This error is difficult to quantify. We describe an experimental setup for estimating the angular error distribution in Section 3.4.3.

The distribution of the scanning errors due to poor synchronization remains relatively constant as long as the sensor is stationary. The effect of them on the image may be predicted and appropriate corrections can be made by simple processing of the range images. For example, ignoring the first few rows would eliminate most of the unreliable measurements. However, additional image *distortion* may occur in some applications. In autonomous navigation applications, for example, the sensor may be subject to motion, shock, and vibrations. This may produce large errors in scanning angles because of the mirrors getting out of phase. Those errors are hard to quantify. This effect is unavoidable except through the use of an external stabilization device.

The distribution of the scanning errors due to poor synchronization remains relatively constant as long as the sensor is stationary. However, additional image *distortion* may occur in some applications in which the scanner is subject to external motion.

Figure 3.9: Synchronization error in Perceptron image

The black wire-frame rectangle marks the correct position.

Localization Even with perfect synchronization and no distortion due to external motion, sensor geometry may be incorrect if one uses a simplified model of the scanning mechanism. The ideal geometric model of Section 3.2.2 assumes implicitly that all the directions of measurement originate from a single point. This is equivalent to saying that the mirrors are infinitely small and are located exactly at the same point. some dimension and do not coincide in space. Therefore, the measurement directions do not intersect at a single point. In realisty, the measurements intersect along a line instead of intersecting at a point. This is a problem of *localization* of the origin of the sensor which translates into a systematic error in the conversion from (r, θ, φ) to (x, y, z). It could be solved by modeling the scanning mechanism or by calibration. For medium-range scanners such as Perceptron and Erim, the localization error is small compared to the errors on range and scanning angles.

3.3.3 Range/Intensity Crosstalk

Ideally, we would like the range measurements to be completely independent of the reflective properties of the observed object. Unfortunately, they do influence the range measurements and can even render range useless in some cases. This *crosstalk* effect between range and intensity is due to a number of causes.

The first cause for the crosstalk effect is a fundamental property of direct am-cw range measurement. The standard deviation given by Eq. 3.6 depends on the reflectance of the observed material. Roughly speaking, the lower the intensity, the higher the range noise. This affects only the variance of the measurement, not its mean value.

Another source of crosstalk is in the implementation of the detection electronics. Typically, the receiver electronics operate optimally only in a narrow range of intensities compared to the large dynamic range of intensities that can be observed. As a result, surfaces that reflect intensities outside of the optimal operating range will produce noisy or even erroneous range measurements. This effect can be reduced by dynamically adjusting the operating range according to the intensity. The Perceptron scanner implements such a solution. However, the low intensities (dropout) and the high intensities (saturation) still produce spurious range readings. There are ways to increase the dynamic range of the receiver but they are not implemented in most scanners available to date.

The crosstalk effect becomes more noticeable at edges or on textured surfaces. In those cases, the beam illuminates a region that contains points of different surface reflectance. Since those points may generate slightly different range measurements, the situation is similar to the one discussed in Section 3.3.1 except that the mixing is due to the variation of reflectance within the spot instead of the variation of range. A consequence of this effect is that the behavior of the range measurement at the edge between two surfaces with different reflectance may be unpredictable.

The crosstalk problem cannot be completely eliminated. However, some additions to the basic sensor design can diminish its effects. For example, the Perceptron scanner adjusts dynamically the operating range of the receiver, and uses a lookup table built using an off-line calibration procedure to correct the range as a function of intensity.

Figures 3.10 to 3.12 illustrate the crosstalk effect. In order to quantify the crosstalk effect, we designed an experiment in which a target with low reflectance is observed against a background of higher reflectance (Figure 3.10). Considering one scanline in the range image, the dark target is located between columns 121 and 135. We computed the mean and variance of the range and intensity distributions at each pixel in the scanline by taking 100 images of the scene. Figure 3.11 shows the mean values as a function of the column number. The mean intensity drops sharply at the edges of the black target and remains at a low level between them, as expected. The mean range remains roughly constant except for a sharp discontinuity at each edge. The reason is that the intensity from both materials is mixed at the edges, therefore the range is not properly corrected. Figure 3.12 shows the variance of the range and intensity distributions. This clearly shows a sharp increase in σ^2 between the high intensity background and the black target, as expected from the theoretical expression of range noise.

3.3.4 Range Drift

We observed a significant drift of range measurements over time. To illustrate this effect, we placed a target 6 m from the origin of the Perceptron scanner and acquired one image per minute over 24 hours, during which the scene was static.

Figure 3.13 plots the sensed range at one target pixel. The figure shows a dramatic variation over time. Between hours 0 and 3, the ranges climb approximately one meter, as if the sensor were translating away from the target. After this four hour "warm-up" period, the sensed ranges reach a plateau where they remain, with apparently random variations, for the rest of the day.

We hypothesized that some of the variations might be due to temperature changes. To test this hypothesis, we placed an electric heater directly behind the scanner, and repeated the above trial, acquiring images at 2 Hz over two hours (14,000 images). Without the heater, the temperature was $21°C$, and we observed approximately constant range measurements. We turned on the heater and after 30 minutes the temperature climbed to $45°C$. During this time, the sensed ranges fell about 40 cm, a substantial decline. When we turned off the heater, the sensed ranges gradually increased until they regained their original level. This demonstrates conclusively that temperature changes cause the distribution of range values to translate by significant amounts.

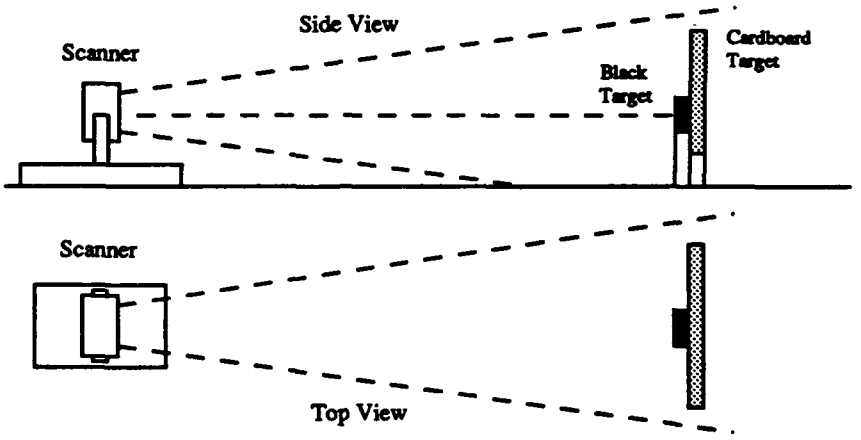


Figure 3.10: Experimental setup to study range/intensity crosstalk

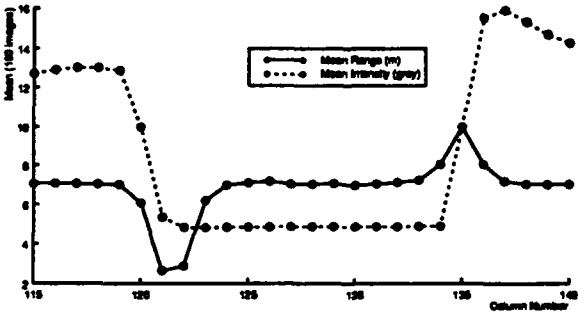


Figure 3.11: Mean range and intensity for black and white targets

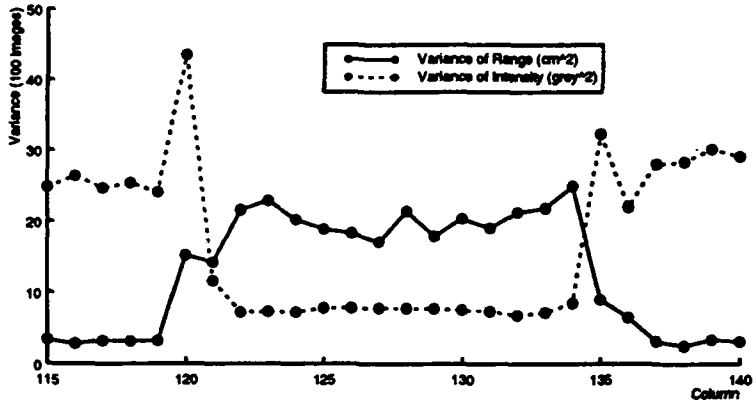


Figure 3.12: Variance of range and intensity for black and white targets

It is clear that heat cannot directly affect the phase shift which carries the range information. Therefore, the drift observed in those experiments is due to poor temperature compensation in the electronics used in the scanner. Improvements in the electronics have reduced this effect considerably. An important lesson is that such effects may dominate the errors due to the physics of the measurements.

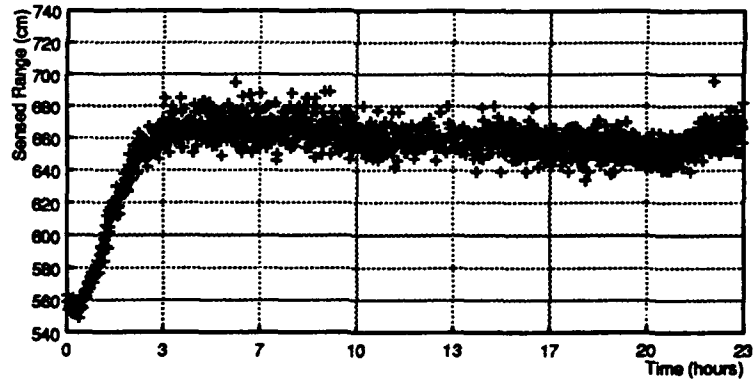


Figure 3.13: Range measurements vary over time

The target is a sheet of cardboard 6m in front of the scanner.

3.4 Accuracy and Precision

We have introduced a theoretical framework in Section 3.2.3 that leads to a characterization of expected sensor accuracy for am-cw laser radars. However, it is important to verify that the sensors do indeed follow the theoretical model. In particular, real sensors include effects such as those described in Section 3.3 that are not part of the theoretical framework. Actual sensor accuracy is important in determining what algorithms and what applications

	a (deg)	r_0 (m)	Error (m)
Black (Sunny, lights)	45.59	-1.32	0.28
Black (Cloudy, lights)	44.34	-0.73	0.20
Black (Cloudy, no lights)	44.11	-0.81	0.29
Cardboard (Cloudy, lights)	44.92	-1.11	0.12
Wood (Cloudy, lights)	45.57	-1.26	0.09

Table 3.2: Accuracy results for different targets and lighting conditions

The table shows accuracy results for various combinations of targets (one untreated cardboard slab, one cardboard slab painted black, and a planar piece of wood) and lighting conditions (sunny, cloudy, with and without room lights).

are appropriate for a given sensor. In this section, we describe a series of experiments designed to measure range accuracy for the Erim and Perceptron sensors under different conditions and to compare it with the predicted theoretical values. Following [1], we distinguish between *accuracy*, the difference between measured range and actual range, and *precision*, the variation of measured range to a given target. To separate the errors due to scanning and the errors due to actual range measurements, we distinguish between range precision and angular precision.

3.4.1 Accuracy

To determine the accuracy of the range measurements is to identify the distance between them and ground truth ranges. For a target point lying in the direction (φ, θ), let $r_{\varphi,\theta}$ be the range measurement reported by the scanner and let $d_{\varphi,\theta}$ be the true distance from the geometric origin, measured with a tape measure. Under ideal conditions, we expect to observe a linear relationship:

$$r_{\varphi,\theta} = a d_{\varphi,\theta} + r_0 \tag{3.7}$$

where r_0 is the *offset distance* from the origin to the (conceptual) surface corresponding to a range measurement of zero and a is the slope.

To determine the parameters a and r_0, we acquire range measurements of targets at six known distances between 6 and 16 m, and fit a line to the data. We illustrate the results for the Perceptron scanner in Table 3.2, which shows the extracted parameters from five trials under different conditions. Because of range drift (cf. Section 3.3.4), we do not assign high confidence to the particular slope, intercept, and rms error entries in the table.

Nevertheless, the variation with surface material and lighting conditions is obvious. This suggests that the accuracy of the scanner depends significantly on variables in Eq. 3.7, including surface material, ambient illumination, and temperature. It also suggest that the effect of those variables is amplified by the particular hardware used in those sensors, as described in Section 3.3. We conclude by remarking that preliminary analysis of this and other data suggests that the accuracy does not depend significantly on target distance.

3.4.2 Range Precision

To determine the precision of the range measurements is to identify by how much repeated range measurements vary. Here, we quantify the precision as the standard deviation of a distribution of measurements.

We have conducted a number of experiments in which we take 100 images at each target position, and compute the standard deviation of the depth measurements. In the experiments, we have examined how precision changes as a

function of ambient illumination conditions, surface material of the target, distance from the scanner to the target, and beam incidence angle at the target. In this section, we report on the effect of ambient illumination for the Perceptron scanner, and refer readers interested in the other properties to Appendix A of [10].

To study the effect of ambient light on sensed range, we place a target (in this experiment, a cardboard slab painted black) at a known distance, take 100 images, and compute the variance in range at particular pixels. We repeat this procedure for six target distances between 6 and 16 m under different indoor lighting conditions.

Figure 3.3 plots the results, which show that the precision decreases with intensity of illumination. The results strongly support the conclusion that the brighter is the ambient light, the larger are the temporal variations in the range measurements. As in the case of accuracy, above, the effect of the variables in Eq. 3.6 is clearly visible in those experiments, and it is amplified by the particular hardware implementation used. The results also illustrate the dependence of precision of the square of the target distance (cf. Eq. 3.6).

3.4.3 Angular Precision

To measure the angular precision of a scanner, or its repeatability in pixel position, we fix the scanner configuration and scene, and take a series of images. Naturally, we expect a static point in the scene to project to one and only one pixel position in each of the images. We position the scanner in front of a vertical wall on which we have drawn white circles of radius 12 cm, surrounded by black squares. We acquire a sequence of 10^4 images. From each intensity image we extract the white circle by thresholding, and then compute its centroid. We compute the standard deviations of these centroids for several row and column positions.

We find that for both scanners, the standard deviation of the centroid is an order of magnitude smaller than the nominal horizontal and vertical angle increments, varies little over time, and varies little over different pixels in the image. The Perceptron has significantly better angular precision than the Erim.

These findings suggest that the angular precision of the scanners is not a limiting factor. However, the experimental setting—stationary sensor and scene, gathering information over a region—represents a best case, for which the angular precision should be zero. Thus, the findings do not reveal the limitations introduced by relative sensor motion, and do not justify neglecting angular variations as a source of random disturbances in the measurement process.

3.5 Discussion

In this paper, we have examined in detail the 3-D measurements supplied by amplitude-modulated laser radars. We presented measurement models for this class of sensors, identified problems unique to the technology, and problems specific to the implementation of sensors currently available to the robotics community, presented experimental performance results, and related our practical experience with the sensors.

How good are the three-dimensional measurements? In terms of speed and reliability for medium-range operations, we are not aware of any sensors with superior performance. So the short answer is that they are very good.

The special problems—mixed pixels, crosstalk, deviations from the scanning pattern, and range drift—make it necessary to pre-process the images. For some problems, e.g. mixed pixels, no solutions exist. Thus, range data interpretation algorithms, like so many others in machine perception, must tolerate spurious data.

The quantitative performance, in terms of accuracy and precision, is highly variable. It depends on geometric factors such as incidence angle and target distance. We understand these reasonably well. But it also depends significantly on non-geometric factors such as temperature, ambient illumination, and surface material type. Even though detailed measurement model which include some of those variables have been developed, there is still a significant discrepancy between the theoretical sensor characteristics and the observed performance.

3.6 References

[1] P. Besl. Active, Optical Range Imaging Sensors. *Machine Vision and Applications*, 1:127–152, 1988.

[2] R. Dunlay and D. Morgenthaler. Obstacle Detection and Avoidance from Range Data. In *Proc. SPIE Mobile Robots Conference*, Cambridge, Massachusetts, 1986.

[3] Environmental Research Institute of Michigan. *Proc. Range and Reflectance Processing Workshop*, Ann Arbor, Michigan, 1987. Limited distribution.

[4] H. Everett. Survey of Collision Avoidance and Ranging Sensors for Mobile Robots. *Robotics and Autonomous Systems*, 5:5–67, 1989.

[5] M. Hebert and T. Kanade. 3-D Vision for Outdoor Navigation by an Autonomous Vehicle. In *Proc. Image Understanding Workshop*, Cambridge, Massachusetts, 1988.

[6] M. Hebert, T. Kanade, and I. Kweon. 3-D Vision Techniques for Autonomous Vehicles. Technical Report CMU-RI-TR-88-12, Robotics Institute, Carnegie Mellon University, August 1988.

[7] M. Hebert, E. Krotkov, and T. Kanade. A Perception System for a Planetary Explorer. In *Proc. IEEE Conf. on Decision and Control*, pages 1151–1156, Tampa, December 1989.

[8] R. Jain and A. Jain. *Analysis and Interpretation of Range Images*. Springer-Verlag, New York, 1990.

[9] E. Krotkov. Laser Rangefinder Calibration for a Walking Robot. Technical Report CMU-RI-TR-90-30, Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, December 1990.

[10] I. Kweon. *Modeling Rugged Terrain with Multiple Sensors*. PhD thesis, School of Computer Science, Carnegie Mellon University, January 1991.

[11] I. Kweon, R. Hoffman, and E. Krotkov. Experimental Characterization of the Perceptron Laser Rangefinder. Technical Report CMU-RI-TR-91-1, Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, January 1991.

[12] D. Nitzan. Assessment of Robotic Sensors. In *Proc. Intl. Conf. Robot Vision and Sensory Controls*, pages 1–11, London, England, April 1981.

[13] D. Nitzan, A. Brain, and R. Duda. The Measurement and Use of Registered Reflectance and Range Data in Scene Analysis. *Proc. IEEE*, 65(2):206–220, February 1977.

[14] Odetics, Inc., Anaheim, California. *3D Laser Imaging System*, 1989.

[15] Perceptron, Inc., Farmington Hills, Michigan. *LIDAR Scanning System*. U.S. Patent Appl. No. 4226-00015.

[16] W. Pratt. *Laser Communication Systems*. Wiley, 1969.

[17] P. Veatch and L. Davis. Efficient Algorithms for Obstacle Detection Using Range Data. *Computer Vision, Graphics, and Image Processing*, 50:50–74, 1990.

[18] R. Watts, F. Pont, and D. Zuk. Characterization of the ERIM/ALV Sensor—Range and Reflectance. Technical Report , ERIM, Ann Arbor, Michigan, 1987.

[19] D. Zuk and M. Delleva. Three-Dimensional Vision System for the ASV. Final report no. 170400-3-f, DARPA, Defense Supply Service-Washington, January 1983.

[20] D. Zuk, F. Pont, R. Franklin, and V. Larrowe. A System for Autonomous Land Navigation. Technical Report IR-85-540, ERIM, Ann Arbor, Michigan, 1985.

Chapter 4

Progress in Neural Network-Based Vision for Autonomous Robot Driving

4.1 Introduction

The ability of an artificial neural network to perform a task is heavily influenced by the quality of its training set. If the training set contains examples taken from the full range of situations the network is expected to handle, the network will learn to perform the task accurately. But if important situations are missing during training, the network is likely to perform poorly when it later encounters the novel circumstances[1].

In the domain of autonomous driving, we have shown that the connectionist architecture shown in Figure 4.1 can quickly learn to steer by watching a person drive [2]. The network receives live input from a camera on the vehicle. The network is trained using back-propagation [3] to activate the output unit representing the driver's current steering direction. After approximately four minutes of watching a person drive on a particular type of road, the network is able to take over for itself and drive at up to 55 miles per hour. Individual networks have been trained to drive in a wide variety of situations, including single and multi-lane roads with and without lane markings.

4.2 Transitory Feature Problem

Certain types of transitory driving situations have proven troublesome for this connectionist approach to autonomous driving. Two examples of temporary but problematic situations are illustrated in the real video images of Figure 4.2. The left image shows a typical multi-lane highway scene used to train the network. It has features including the lane markings down the left, right and center of the road, and a patch of grass from the median in the upper left corner. The other two images illustrate deviations from this typical scene. The center image shows a jersey barrier instead of a patch of grass in the upper left corner as the vehicle drives over a bridge. The right image shows the vehicle passing another car.

The reason this type of transitory disturbance causes trouble is that the network is trained over a relatively short stretch of road (< 2 miles). As a result, during training the network is not exposed to all the possible driving situations it might encounter when driving autonomously. In particular, since situations like the two illustrated in Figure 4.2

[1] An earlier version of this paper first appeared as Progress in Neural Network-based Vision for Autonomous Robot Driving, by D. Pomerleau, in Proceedings of the 1992 Intelligent Vehicles Symposium

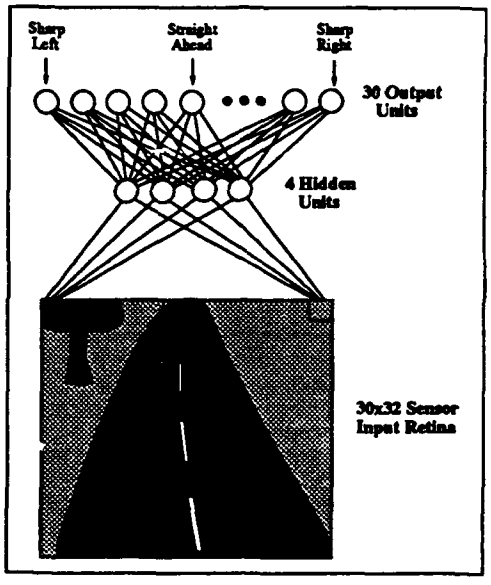


Figure 4.1: Neural network architecture for autonomous driving.

are relatively rare and limited in duration, even if the network sees them during training, it doesn't learn enough from its brief exposure to handle them appropriately. As a result, while the network is capable of reliably driving in situations closely resembling those it was trained on, when it encounters novel situations like the ones illustrated above, it frequently steers incorrectly.

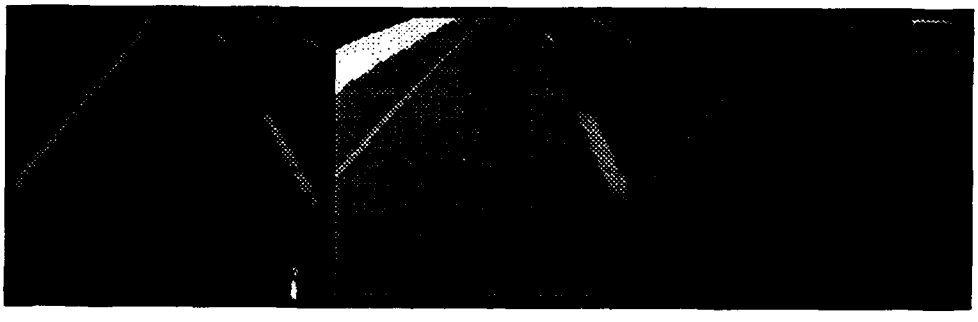


Figure 4.2: Three video images of a multi-lane highway.

The influence spurious image features can have on driving performance can be seen in Figures 4.3. The left image in Figure 4.3 illustrates a typical reduced resolution multi-lane highway image like the ones ALVINN was trained on. The dark triangle in the upper left is the green grass on the left side of the road. Notice that ALVINN's output response is nearly identical in position to the target, indicating ALVINN is steering in the correct direction. The image on the right of Figure 4.3 has a guardrail on the left side, which appears as a white stripe in the upper left corner. Notice ALVINN's steering direction is significantly disturbed by this relatively minor change to the image.

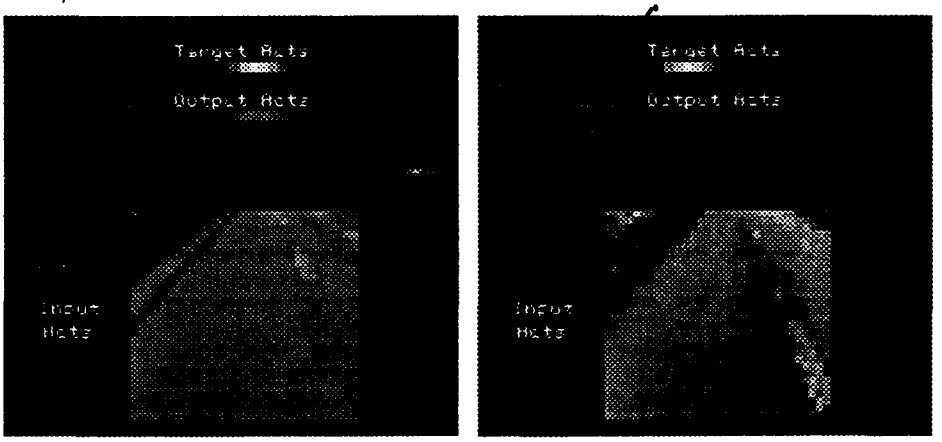


Figure 4.3: Two low resolution images of a multi-lane highway and the network's response. The image on the left is a typical highway image, with no unexpected features. As a result, the network responds correctly. The image on the right contains a guardrail on left side of the road (the white stripe in the upper left corner). Since the network did not see a situation like this during training, its steering response is far from correct.

The reason for this disturbance is illustrated in Figure 4.4. Each of the rectangles labeled "Input-Hidden1 Weights" through "Input-Hidden4 Weights" represents the weights projecting from the input retina to one of the four hidden units in the network. White squares within each of these rectangles represent excitatory weights, black squares represent inhibitory weights, and grey squares represent weights with small magnitude. The connections from the pixels in the upper corners of the image to each of the hidden units have relatively large magnitude, indicating the network is using those pixels to determine the correct steering direction. The reason the network came to rely heavily on those pixels is that during training, no guardrails or passing cars appeared in the periphery during training. In the case of the left periphery, it contained only grass over the entire training sequence. As a result, the size of the grass patch was a good indication of how sharply the vehicle should turn. The larger the patch, the more towards the left the vehicle was, and therefore the more the vehicle should turn to the right. Because of the high correlation between the size of the dark patch and the correct steering direction, the network learned to use this feature to determine how to steer. Therefore, the connections from the upper left pixels were given large weights, while the center and bottom portions of the image were largely ignored. The same argument hold true for the pixels in the upper right corner: the network relies heavily on them because of their consistency during training.

The quantitative effect transitory image features like guardrails have on driving performance is shown in the leftmost two bars of the graph in Figure 4.5. These two bars represent the performance of a network trained without noise on a sequence of multi-lane images like the ones shown in Figure 4.2. The leftmost bar shows the network's average steering error on a disjoint set of 180 multi-lane road images. The steering error represents the curvature difference between the steering arc suggested by the network for an image and the arc the person was driving along when the image was taken. The bar next to it, labeled "Images w/ Guardrail" shows the average steering error of the same network on a 32 image subset from the test sequence which contained a feature not present in the training images, namely a guardrail on the left side of the road. The network's steering error more than doubles on images containing the novel feature, clearly illustrating the detrimental impact these features can have on performance. If allowed to steer autonomously over the stretch of road where the guardrail images were taken, the network would have steered

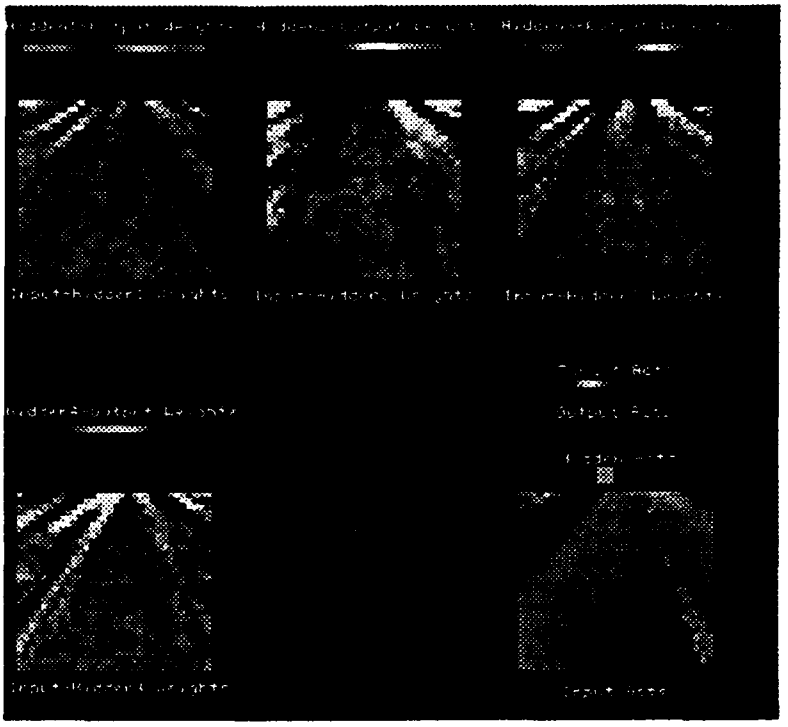


Figure 4.4: Weight diagram of a network trained without noise.

off the road.

The problem caused by transitory image features results from insufficient diversity in the training examples. The network does not encounter these transitory features frequently enough during the short training period to learn to ignore them. One useful aspect of the problem, exploited in the solution below, is that these transitory features do not radically alter the overall appearance of the image. Due to the redundancy of features in the image, if the network could learn to ignore spurious features, it should be capable of driving accurately.

4.3 Training with Gaussian Noise

A commonly employed technique for improving generalization from a limited amount of training data is to add uncorrelated gaussian noise to the training patterns [4]. The idea is that adding noise to the input prevents the network from relying on idiosyncrasies in the training patterns to perform the task. Sietsma and Dow found a dramatic improvement in generalization when noise was added to the training patterns on a frequency classification problem. In their task, the input was a 64 unit vector whose input activation pattern formed a sine wave of a particular frequency. The task was to classify an input sine wave according to its frequency, regardless of its phase within the input field. They found that when gaussian noise was added to the training patterns, the resulting network made dramatically fewer classification errors on novel, noise-free input patterns (0.5% vs. 17%).

We performed a similar experiment by adding various amounts of gaussian noise to the road images used for

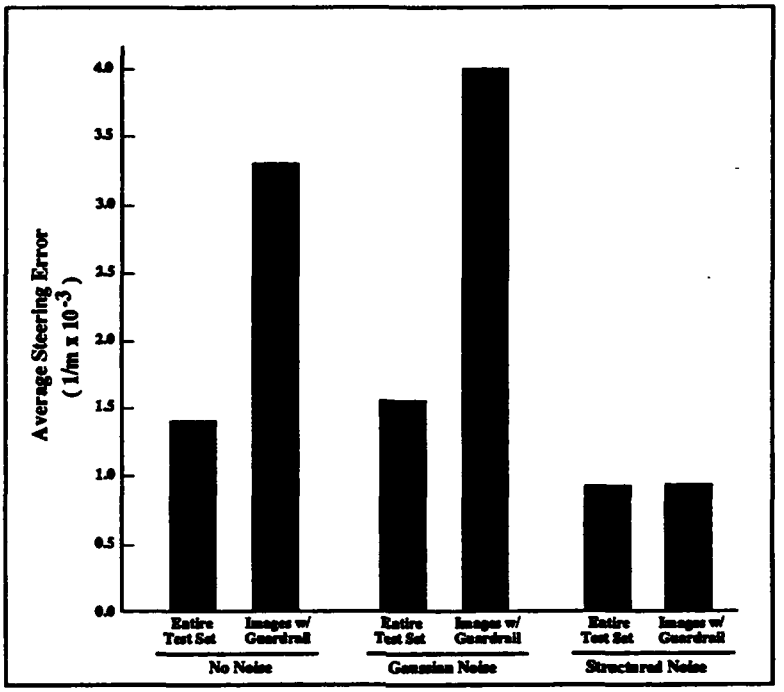


Figure 4.5: Graph illustrating the performance of networks trained using three different techniques on a set of 180 images, and on a subset of 32 images containing a guardrail.

training. The noise had a mean of 0 and a standard deviation ranging from 0.4 to 1.2 (The pixel intensity values ranged from -1.0 to 1.0). Figure 4.6 shows one corrupted road image used for training.

Surprisingly, networks trained with this noisy input performed uniformly worse than network trained without noise, as illustrated by the bars in Figure 4.5 labeled "Gaussian Noise". They represent the *best* performance of any of the networks trained with gaussian noise. Both on the entire test set, and particularly on the guardrail subset of images, the networks trained with noise steered less accurately than the network trained without noise.

The reason for this drop in performance is evident in Figure 4.6. The finer image features such as the lane markings, which were visible in the noise-free image of Figure 4.3, have been obscured by the noise. The only visible feature is the patch of grass in the upper left corner. The network had no choice but to key on the size and location of the grass patch to determine the steering direction. This degraded performance on the test set because despite its large size, the grass patch is actually a less reliable feature than the lane markings, since it is frequently obscured by guardrails and jersey barriers.

Gaussian noise is a poor model of the important "noise" that occurs in the input while driving. The noise that matters results from the appearance or disappearance of coherent 2-D features such as guardrails and other cars. Modeling these irrelevant features and adding them to the input while training can improve generalization dramatically, as illustrated by the last two bars in Figure 4.5. Networks trained by adding structured noise to the input, as described in the next section, generalized better on the test set as a whole than networks trained without noise or with gaussian noise. Even more significant was the performance improvement of the network trained with structured noise on images with spurious features in them, as illustrated by the low error of the network on the guardrail images. In fact, the

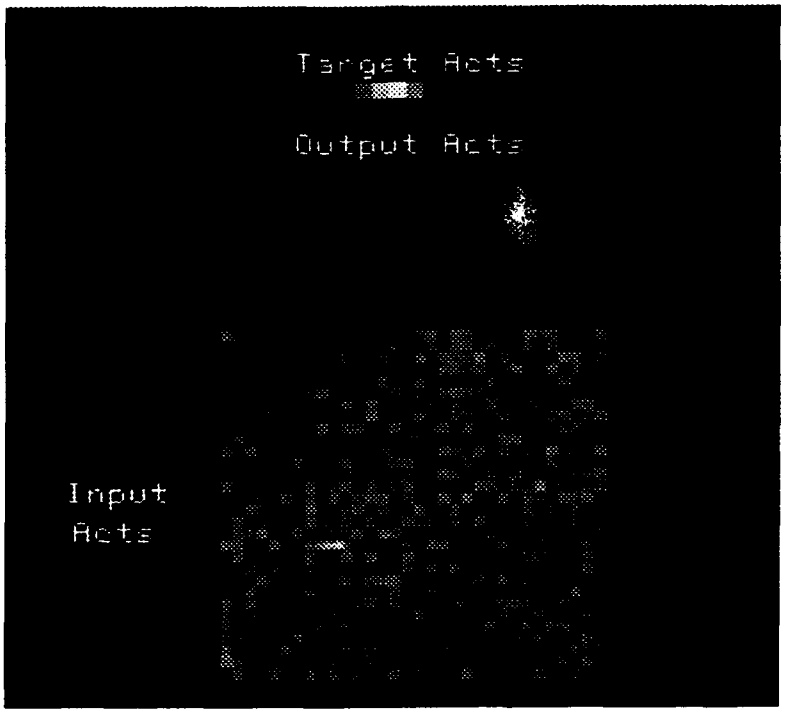


Figure 4.6: A multi-lane highway image corrupted with gaussian noise.

network's steering performance on the novel guardrail images was not significantly different than its performance on the test set as a whole, demonstrating that it has learned to ignore spurious image features (See Figure 4.5).

4.4 Characteristics of Structured Noise

A number of straightforward characteristics of structured image noise can be employed to improve network generalization. The most obvious features is the high degree of spatial correlation. Important noise does not appear as corruption of random image pixels. Instead, the physical object (like a car or guardrail) causing the noise makes a 2-D projection into the input image. As a result, the important image noise takes the form of a two-dimensional regions of nearly uniform intensity. While objects with multiple or variable intensities do occur, their rarity makes a uniform intensity model a reasonable approximation.

A more subtle characteristic of structured image noise results from the fact that objects can change in three ways. Objects can suddenly appear in the image, obscuring part or all of a previously visible feature. An example of this effect is when a car passes and obscures the lane markings. Objects can also disappear from the image, making previously hidden features visible. This effect might occur when the guardrail disappears, revealing the grass on the other side. Finally, a feature can change color or brightness, as when the centerline changes from white to yellow. Shape is not considered a changing feature characteristic, since objects visible when driving are assumed to be rigid.

Another useful domain-specific characteristic of structured image noise is that when driving, irrelevant features

are more likely to occur in the periphery of the image than in the center. Even if during training the appearance of the terrain off to the side of the road remains constant, it is helpful for the network to learn that the appearance of the periphery can change dramatically. This way the network learns not to rely on the position of the guardrail or the size of the grass area in the off-road region, since they may be obscured or disappear at any time.

The size of significant image features can be constrained by estimating the size and distance to noise features. The retinal size of significant features ranges from a few tens of pixels for the lane markings or guardrail to several hundred pixels for large objects such as passing trucks. The *shape* of significant image features is also constrained in the domain of autonomous driving. Image features are frequently oriented with their primary axis pointed towards the vanishing point of the image. The technique for incorporating these characteristics of image features when generating noise is described in the next section.

4.5 Training with Structured Noise

Structured noise characteristics are used during training to determine the appearance of the noise to be added to the input patterns. Instead of adding gaussian noise to each pixel, the following technique is employed to add or remove coherent two-dimensional features from the training patterns.

First, for each pattern on each epoch of the back-propagation, a decision is made whether to add noise to that pattern or not. Empirically, adding noise on 3 out of every 4 presentations of a pattern seems to be a good compromise between teaching the network to be insensitive to noise and teaching it to process clean images correctly.

Once the decision is made to add noise to a pattern, the next question is where to put it. We have found that when trained on images with a single noise feature, a network is able to generalize to images with multiple noise features. Therefore, at most one noise feature is added per image during training. The location for this single noise feature is selected randomly with a bias towards the periphery of the scene, corresponding to the upper corners of the image. That is, the likelihood that a pixel will be chosen as the starting point for the noise feature is proportional to its proximity to one of the upper corners of the image. This periphery bias roughly models the tendency of noise features to appear away from the path directly ahead of the vehicle.

After determining the starting location for the noise feature, a decision is made whether a new feature should be added or an existing feature should be removed from that position. This choice is made randomly, but with a bias towards deleting a feature if one exists at that location, and towards adding a feature if that location appears to be "feature free". The judgement concerning whether a feature exists at a location is made based on the size of the uniform region that location is part of. First, a region is grown around the chosen location to encompass all the contiguous pixels whose intensity is within a fixed threshold of the chosen pixel's value. If the size of this region is within the size range of interesting features as characterized above, then the location is considered to be part of a feature and that feature is removed. If the size of the region falls outside the size range of interesting features, it is considered part of the background, and a new feature is added at that location.

Removing a feature is easy. The pixels defining the feature have already been determined in the region growing step described above. Deleting the feature involves changing the values of all the pixels within the region to a new randomly chosen intensity. Altering a region's intensity models the corresponding object changing color. By selecting the new intensity to be the same as the intensity of the area surrounding the feature, the disappearance of the feature can also be simulated. The result of using this technique to alter an existing feature is illustrated in the left image of Figure 4.7. The image is identical to the one on the left of Figure 4.3, except that the patch of grass in the upper left corner has changed intensity from very dark to very light.

Adding a new feature to model the sudden appearance of objects such as a guardrail or automobile is more difficult, since unlike in the feature alteration process described above, the shape of the feature is not known. The

shape of spurious 2-D image features can in theory be arbitrary. There is the weak constraint that significant features tend to have their major axis pointing towards the vanishing point. However, this orientation specificity is difficult to implement directly in the noise generation model since it requires knowledge of the sensor geometry.

A simpler technique for generating reasonable spurious features involves using the shape of the feature detectors the network develops in its internal representation to bias the shape of the simulated noise features. As is evident in the weights from the input retina to the hidden units in Figure 4.4, the network develops a strong model of the shape of important image features. The network's knowledge that features tend to be oriented towards the vanishing point is demonstrated by the tendency of features in the receptive fields of the hidden units to converge towards the top.

To bias the new feature's shape using the shapes of the features in the network's internal representation, a random hidden unit is first selected. The values of the weights from the input retina to this hidden unit are then used as the "image" in which to grow the new feature. In other words, a pixel in the vicinity of the feature's start pixel is chosen to be an element of the feature if the weight of the connection from it to the chosen hidden unit is sufficiently similar in value to the weight of the connection from the feature's start pixel to the chosen hidden unit. By biasing the shape of noise features by the shape of important features in the internal representation, this technique ensures that only noise features with a reasonable shape are added to the input.

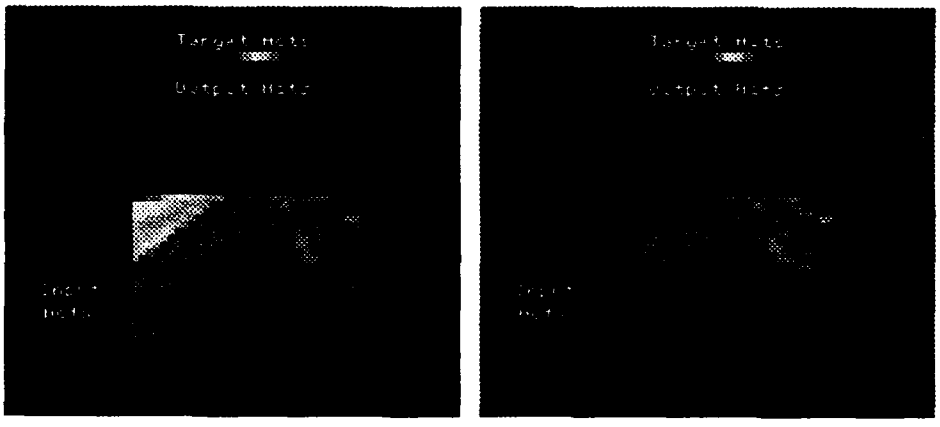


Figure 4.7: Two images augmented with structured noise. In the left image, an existing feature (the patch of grass in the upper left corner) has been altered by changing its intensity. In the image on the right, a new dark feature has been added on the right.

To mimic situations in which image noise does not precisely align with significant feature detectors, a spatial coherence constraint is added to the feature growing algorithm. The following equation balances the tendency of the feature growing algorithm to follow hidden unit weight contours with the tendency to create a spatially compact feature. A pixel i will be included in a newly created features if:

$$\alpha|w_{ih} - w_{sh}| + (1 - \alpha)|i - s| > T$$

In this equation, w_{ih} is the weight value of the connection originating at pixel i in the input image and terminating at the chosen hidden unit h. The variable w_{sh} corresponds to the weight of the connection originating at s, the pixel chosen as the start position of the feature, to the chosen hidden unit h. The quantity $|i - s|$ represents the Euclidean distance between pixels i and s in the input retina. The constant α is used to weight the tendency to follow hidden unit weight contours with the tendency to keep the feature spatially compact. With an α of 0, the new feature will form a circular

region centered on pixel s. With an α of 1, the new feature will grow to include all the pixels in the image having connections to hidden unit h with a weight close in magnitude to the connection from pixel s to unit h. Randomly choosing an α from the range 0.1 to 0.3 for each image creates realistic looking new noise features. The threshold T is used to limit the growth of the new feature. If the weight from pixel i to hidden unit h differs significantly from the weight from the start pixel s to unit h, or if pixel i is a great distance from pixel s, then the threshold T will be exceeded by the above sum and pixel i will not be included in the new feature.

Once the position and shape of the noise feature is determined, it is made to contrast with the surrounding area by filling it with a randomly chosen uniform brightness. A real video image in which a noise feature (the dark region in the upper right) has been added is shown in the right image of Figure 4.7. Notice how the feature appears close to the upper corner of the image due to the periphery bias built into the noise feature generation algorithm. The right image of Figure 4.7 also illustrates how growing noise features along important contours in the hidden unit representation results in features with appropriate orientations for the domain. In this example, this bias results in a feature oriented diagonally towards the vanishing point. The feature is spatially compact due to the bias in the noise generation algorithm towards choosing pixels in the vicinity of the feature's start pixel. This combination of biases based on known, consistent image feature characteristics results in the generation of noise features with realistic appearance. In fact, the feature in right image of Figure 4.7 appears very much like a car passing by on the right side of the vehicle, as in the right image of Figure 4.2.

As illustrated in the bar graph of Figure 4.5, a network trained by adding and removing features steers more accurately than a network trained without noise, particularly on images containing spurious features. The reason for this is evident in the weight diagram of Figure 4.8. The network shown in this diagram was trained on the same sequence of images as the network shown in Figure 4.4, except that on each iteration of back-propagation, 75% of the patterns were randomly selected to have a noise feature added to their input. Varying the noise at every epoch prevents the network from learning characteristics of a single noise feature. The remaining 25% of the patterns were presented without noise to ensure the network would also handle noise free situations.

It is clear by looking in the upper corners of the input-to-hidden weight arrays that adding structured noise to the image has the desired effect. Namely, the network learned to rely less on features in the periphery than the network depicted in Figure 4.4. The network developed detectors primarily for the line marking the left lane boundary, since this is the feature that appears most reliably in this type of road image. Because of its frequent occlusion and absence from the image, the dashed line marking the right boundary of the lane was given little importance in the internal representation. The resulting performance improvement is illustrated by the input pattern and corresponding network response in the lower right corner of Figure 4.8. The input pattern is the same guardrail image that confused the network trained without noise on the right side of Figure 4.3. This network handles the guardrail image perfectly, since it has learned not to be disturbed by features in the periphery.

4.6 Improvement from Structured Noise Training

The bar graph in Figure 4.5 illustrates the significant increase in steering accuracy which results when structured noise is added in the training process. This steering accuracy improvement translates directly into dramatic gains in driving performance. Quantitatively, we have found that a network trained without adding structured noise on a two mile stretch of a four lane divided highway was capable of driving autonomously for only about four miles before straying from the road because of a spurious feature. Driving successfully for even this relatively short distance was somewhat fortuitous, in that it was achieved on a stretch of road free from guardrails and other potentially confusing permanent features, and at a time of day when there was very little traffic to confuse the network. When other vehicles did appear, the network trained without noise frequently swerved towards or away from them depending on their brightness relative to the background. The swerves were relatively small, so the safety driver allowed the run to

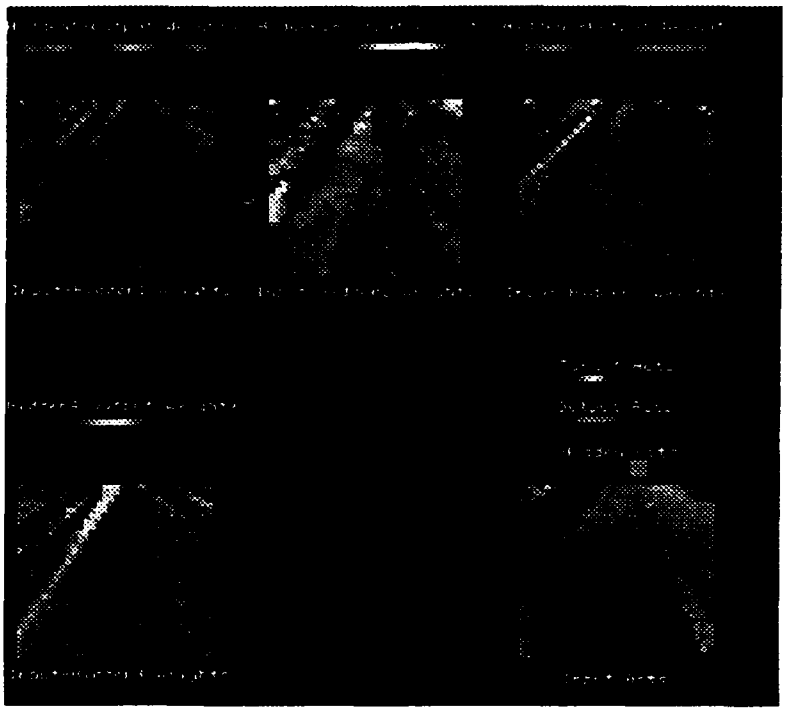


Figure 4.8: Weight diagram of a network trained with structured noise.

continue without interference. The situation which ended the run after four miles was the one depicted in the center image of Figure 4.2. The vehicle encountered a bridge with jersey barriers along the edge of the road. This spurious feature caused the vehicle to swerve dramatically, forcing the safety driver to intervene.

The network trained with structured noise drives significantly better. Its best run was 21.2 miles without human intervention, on the same highway that caused the network trained without noise to fail after 4.0 miles. It made it over the bridge that caused the previous network to fail, and through a number of other situations that would have caused trouble for the network trained without noise. The reason the run came to an end after 21.2 miles was that the width of the road changed significantly, causing the network to become confused.

4.7 Discussion

The appearance or disappearance of irrelevant features can disrupt a network's driving when the network's training did not demonstrate their irrelevance. Adding structured noise to the training patterns using a model describing the characteristics of irrelevant features significantly improves driving performance. A simpler gaussian model of image noise has been shown to be ineffective at compensating for this problem because it does not mimic the important characteristics of real world, structured noise.

But are there other possible solutions which do not require such complex modeling? In theory, training the network over a longer stretch of road than the two miles currently employed should result in a more representative

training set and hence a more robust network. However there are a number of shortcomings to this approach. One long term goal of this work is the development of a "super cruise control system" capable of controlling both the vehicle's speed and steering. If the training period required by this super cruise control is too long, it will be impractical.

But even given unlimited training time, the scarcity of irrelevant features would make it difficult to train a network to ignore them. For instance, over many miles of highway driving, the size of the patch of grass on the left side of the image is a good indicator of the correct steering direction. Only in rare situations, like going over a bridge, will relying on this feature get the network in trouble. Even if the training period were extended to ensure encountering this type of situation, its low frequency would make it beneficial for the network to ignore these few patterns. This is because the network could lower the total error over all training patterns by employing the patch of grass to improve the steering performance on the vast majority of patterns, while suffering substantial error only on the few patterns in which the patch of grass is missing. In other words, the high correlation between an irrelevant feature and the correct output would result in the network employing the feature despite being exposed to a few situations where this degrades performance.

The current model of structured image noise has obvious shortcomings. Noise features do not always occur in the periphery. When driving in traffic, cars directly in front of the vehicle will obscure central image features. But even this simple model is sufficient to demonstrate that dramatic improvements in neural network generalization can be achieved by actively employing domain-specific knowledge during the training process.

4.8 References

[1] D.A. Pomerleau Neural network perception for mobile robot guidance. PhD. Dissertation. Carnegie Mellon technical report CMU-CS-92-115, 1992.

[2] D.A. Pomerleau Efficient Training of Artificial Neural Networks for Autonomous Navigation. *Neural Computation 3:1*, 1991.

[3] D.E. Rumelhart, G.E. Hinton, and R.J. Williams Learning internal representations by error propagation. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. I: Foundations*, 1986.

[4] J. Seitzma, and R. Dow Creating Artificial Neural Networks that Generalize. *Neural Networks, Vol. 4*, 1991.

Chapter 5

Integrating Position Measurement and Image Understanding for Autonomous Vehicle Navigation

5.1 Introduction

Autonomous navigation of an outdoor mobile robot involves many tasks, including road following, obstacle avoidance, landmark recognition, path planning, and navigation. Our test-bed robots, the Navlab and Navlab II, use a variety of sensors and sensor interpretation strategies for these tasks. Typical sensors include video cameras, for road tracking; a scanning laser rangefinder, for detecting obstacles and recognizing landmarks; sonars or giga-hertz radar for quick obstacle detection; shaft encoders and gyroscopes for position tracking[1].

In our first systems, we typically used a single sensor, interpreted by a single module, to control the vehicle for a single task. Individual systems did road following, or path tracking, or obstacle avoidance. As our components became more sophisticated and more reliable, we have combined multiple sensors or multiple interpretation modules for a single sensor to build more capable systems. These systems avoid obstacles while driving along roads, or recognize landmarks while driving, or follow pre-planned paths while checking for obstacles.

Recently we have focused on using maps in combination with more than one sensor interpretation module. The most ambitious missions combine road following, landmark recognition, obstacle detection, and inertial navigation, to drive through an unmodified suburban setting, through intersections and along streets, stopping at a designated destination. We used several different strategies to keep track of vehicle position on the map: dead reckoning kept a running position estimate, perception for road following generated lateral corrections, and landmark recognition generated corrections along the direction of travel.

There were two related problems with our simple position update scheme. First, the corrections were made independently by each module, so the position estimate was completely determined by whichever module had made the most recent correction. Thus, if one of the corrections was slightly incorrect, it influenced the entire system. Second, the position estimates carried no measure of error bounds. The landmark recognition subsystem, in particular, needed predictions from navigation to focus the attention of the sensing and the processing. Without a measure of the error

[1] An earlier version of this paper first appeared as Integrating position measurement and image understanding for autonomous vehicle navigation, by Thorpe C., Amidi O., Gowdy J., Hebert M., and Pomerleau D., in *Proceedings of the Second International Workshop on High Precision Navigation*, Stuttgart, Germany

bounds for a particular position report, the landmark recognizer did not know how far in advance to begin looking, or how large an area to search.

The solution was to use filtered updates, rather than absolute updates. A very simple Kalman filter could easily give approximate bounds on vehicle position, and could be used to integrate new measurements with current estimates, instead of completely throwing out the old position. We have built and tested such a system.

One of the biggest challenges in building our system to filter position updates was calibrating the individual sensing modes. Section 5.2 describes our methods for estimating the error covariances for the laser scanner, dead reckoning, and road following. Another important part of the project was adapting our software structure to make the updates relatively time-independent, and to integrate position information with maps. Our approach to these problems is described in Section 5.3. The actual filter itself, presented in Section 5.4, is clean and simple. Finally, in Section 5.5 of this paper we discuss our experimental results.

5.2 Position Measurement and Error Modeling

5.2.1 Position and Covariances from the Laser Scanner and Object Matching

We use a laser range finder, the ERIM sensor, that measures 64 row by 256 column range images at a rate of two images per second. The range values are coded on 8 bits from 0 to 64 feet yielding a range resolution of three inches. Raster scanning of a scene is achieved by the motion of two mirrors synchronized so that the vertical and horizontal angular steps between consecutive pixels in the image are constant. The vertical and horizontal fields of view are 30 and 80 degrees, respectively. Details about the sensor and its applications may be found in [5, 7, 3]. We focus in this paper on building and using maps of objects detected from the range images, and to manipulate the uncertainty on the locations of the objects.

To detect objects in a range image, we use the algorithms introduced in [4]: First, The 3-D coordinates of each pixel from the range image are computed. Each point is stored in a cell of an horizontal discretized grid. The mean, maximum, and minimum elevations and the surface normal are computed for each cell of the grid and stored. Individual cells are classified as being part of an object based on the similarity of these parameters. Adjacent object cells are grouped into regions, each region corresponding to an individual object. Erroneous initial cell classification might occur, since the parameters are computed at each cell separately. This is corrected by retaining only those regions that are large enough. Spurious objects may still be detected because of noise in the data, for example. Those are eliminated using multiple images of a scene, as described in Sections 5.2.1 and 5.2.1. This technique is simple and efficient and is appropriate for navigation in mild environments where there is a clear separation between terrain and objects.

Uncertainty Modeling

The uncertainty in range measurement may be modeled as a Gaussian distribution of standard deviation σ proportional to the square of the measured range, R. The proportionality factor K depends on the physical characteristics of the sensor, on the reflectivity of the material, ρ, and on the angle of incidence of the beam, α. Although K should vary linearly with ρ, we assume that ρ is nearly constant in our application. This assumption is justified by the fact that the objects used in the experiments reported in this paper were all of similar material. A more accurate model would make use of the intensity reflected by the target to adjust the uncertainty in range. With this simplification, we model the uncertainty as $\sigma_R = K'R^2/\cos\alpha$. K' is computed empirically using the experiments described in [6].

This model is valid only when the uncertainty due to measurement noise dominates the uncertainty due to the sampling of the measurements. This condition is satisfied at the distances that we are considering, typically five to

fifteen meters.

In addition to range uncertainty, we have to take into account the uncertainty in the horizontal and vertical scanning angles, denoted by $\sigma_{(\varphi)}$ and $\sigma_{(\theta)}$ respectively. Typical values of angular uncertainty are 0.1 degrees. Furthermore, we need to convert the uncertainty in (range,angles) space to uncertainty in the (x,y,z) Cartesian space. This can be done in two ways. In the first approach, we observe that the uncertainty on the full measurement $s = (\varphi, \theta, R)$ can be represented by a 3x3 covariance matrix, S. Assuming that range and angles are uncorrelated, the matrix is diagonal, the diagonal elements being $\sigma_{(\varphi)}^2$, $\sigma_{(\theta)}^2$, and $\sigma_{(R)}^2$. We then observe that the Cartesian coordinates $p = (x, y, z)$ of a point are functions of range and angles, $p = F(s)$, where F is a transformation similar to the standard transformation from spherical to Cartesian coordinates. The covariance matrix representing the uncertainty on p is therefore $C = JSJ^t$, where J is the Jacobian of the function F. The matrix J is a function of the scanning angles φ and θ only and can therefore be precomputed at every pixel position in the image.

In the second approach, we define a coordinate system (X,Y,Z) such that Z is aligned with the direction of measurement, and X and Y are in the plane orthogonal to Z. The uncertainty is modeled by a diagonal covariance matrix W. The last element of W is σ_R^2, the first two elements are estimated from the constant uncertainty on scanning angles. The transformation between measurement reference frame (X,Y,Z) and sensor reference frame (x,y,z) is defined by a rotation U, therefore the covariance matrix is given by $C = UWU^t$. As in the first approach, the matrix U is a function of the scanning angles φ and θ only and can therefore be precomputed at every pixel position in the image.

Although the covariance matrix C is a full 3x3 matrix in general, we use only the 2x2 matrix that corresponds to the marginal distribution in x and y. This is because the vehicle model used in our experiments includes vehicle position and heading, but does not include elevation, roll, or pitch. To avoid clumsy notations, the notation C will refer to the 2x2 covariance matrix in the remainder of the section. The uncertainty on the position of a given object is also a 2x2 covariance matrix that is computed from the points detected on the object.

Matching Objects

The main tool for navigation using objects extracted from range images is the matching of objects detected in an image with objects detected in previous images. The problem can be stated as follows: given a current set of objects O at locations $O_i = (x_i, y_i)$, with covariance matrices C_i, and a new set O' at locations $O_i' = (x_i', y_i')$, with covariance matrices C_i', and a transformation between the reference frames of O and O', find the most consistent set of matches between O and O'. The first step is to predict individual matches. The criterion for building the predictions is to declare that object O_i' is a possible match for O_j if $UO_i' + T$ is within the 95by the covariance C_i, where U and T are the rotation and translation part of the translation between the two coordinate frames. This step may generate several possible matches for a given object. A search procedure evaluates the possible combinations of matches and retains the one that is the most consistent. It should be noted, however, that the search is fast because there are only a few ambiguous matches. The ambiguity is limited because of the small number of objects in the scene and because the estimate of the transformation (U,T) is accurate enough to avoid frequent mismatches. The output of the matching process is a set of matches (O_i, O_j'). Using those matches to reduce the uncertainty on the locations of the objects and to update the position estimate is the object of the next two sections.

Map Building

The goal of map building is to combine multiple observations into a consistent map of the environment. In our case, a map consists of a set of objects described by their locations and their associated uncertainty. Using the notations of the previous section, O is the set of objects currently stored in the map, and O' is the set of objects detected in the new image. Matches (O_i, O_j') are computed as described in the previous Section. For each of those matches, the current

estimate of the position object O_i is updated using a standard linear Kalman filter. More precisely, the updated position O_i'' of O_i is given by:

$$O_i'' = O_i + K(UO_j' + T - O_i) \quad K = C_i(C_i + UC_j'U^t)^{-1} \tag{5.1}$$

In this expression, (U,T) is the current estimate of the transformation between map and vehicle position at the time the image was taken, and K is the usual Kalman gain. The covariance matrix is updated by:

$$C_i'' = (C_i^{-1} + UC_j'^{-1}U^t)^{-1} \tag{5.2}$$

Using this type of filtering is beneficial only if several views of each map object are available, in which case the accuracy of object location estimate increases dramatically. In a typical map building run, images are collected as fast as possible thus yielding significant overlap between images. A typical object is visible in five to ten images, thus allowing the filtering process to be effective. A limitation of this approach to map building is the computation time required to process this large amount of data. As a result, map building must be performed off-line or at low vehicle speed.

In addition to refining object locations, map building includes a mechanism for eliminating objects erroneously detected in an image. The detection of those spurious objects may be caused by noise in the input data or mismatches. In addition, moving objects must be eliminated because the system can handle only static environments. This problem is essentially an outlier removal problem and thus cannot be handled by a linear filtering technique such as the Kalman filter. The outlier removal mechanism that we implemented is based on the fact that it is easy to predict which map objects should be visible in an image given the approximate position at which the image is taken, and a model of the field of view of the sensor. We define the confidence of a map object as the number of times it is successfully found in a sequence of images versus the number of times it is predicted to appear in the same sequence. A real object will have a confidence close to one, while an outlier will have a very small confidence since it occurs in only one image in most cases. A moving object would be detected in consecutive range images, but each instance of the object would be too far from the location detected in the previous images to yield successful consistent matches. Thus, the confidence is much smaller than one for moving objects as well. Erroneous objects are therefore eliminated by removing map objects with a confidence lower than a threshold. This confidence mechanism is another reason for using a large number of closely spaced images because the confidence of an object is meaningful only if it has been predicted in a large enough number of frames. Figure 5.1 shows this tracking of objects in ERIM images taken at different vehicle positions.

Position Estimation

Position estimation from range data is the dual of the map building. As before, it involves matching a set O of map objects and a set O' of objects from a new range image. This time, however, the goal is to refine the current estimate of vehicle position given by a rotation U and a translation T defined by the position of the vehicle (x_v, y_v) and its heading θ. We denote by V the vector (x_v, y_v, θ), and we denote by C_v the 3x3 covariance matrix of V. We wish to estimate the discrepancy between the current position estimate and the estimate given by the matches between map and image objects. We denote by ΔV the discrepancy vector $(\Delta x_v, \Delta y_v, \Delta\theta)$, and we denote its covariance by $C_{\Delta v}$. The true rotation between map and current observation is given by $U' = U + \Delta U$, where ΔU is the error in rotation. Assuming that the angular error is small, we can make the approximation $\Delta U = \Delta\theta J$, where J is the derivative of the rotation matrix U with respect to the rotation angle θ. The true translation is given by $T' = T + \Delta T$, where $\Delta T = (\Delta x_v, \Delta y_v)$. Using the same notations as before for the objects, we have: $O_i = U'O_j' + T'$. Therefore, replacing U' by $U + \Delta U$ and T' by $T + \Delta T$, we obtain: $O_i - UO_j' - T = \Delta UO_j' + \Delta T$. Denoting the left-hand side of the equation as P_i and applying the small angle approximation, the relation becomes: $P_i = \Delta\theta J + \Delta T$. Since the right-hand side is a linear function

of ΔV, the relation can be written as $P_i = H\Delta V$, where H is a 2x3 matrix. Since this is a linear relation between the measurement P_i and the estimated variable ΔV, we can directly apply the formalism of the linear Kalman filter. More precisely, if a new estimate of the discrepancy $\Delta V'$ can be computed from the current estimate ΔV given a new match using the relations:

$$\Delta V' = \Delta V + K(P_i - H\Delta V) \quad K = C_{\Delta v}H^t(C_{Pi} + HC_{\Delta v}H^t)^{-1} \quad C'_{\Delta v} = (C_{\Delta v} + H^tC_{Pi}H)^{-1} \tag{5.3}$$

Where C_{Pi} is the covariance matrix of P_i which can be easily computed from C_i, C'_j, and C_v.

The position update could be approached through a different route by observing that the relation $O_i = UO'_j + T$ can is of the form $F(O_i, O'_j, V) = 0$, where F is a non-linear function of V that incorporates the trigonometric functions of θ that appear in U. Viewing (O_i, O'_j) as the measurement, and V as the estimated variable, we can directly apply the extended Kalman Filter formulas [2] to get a new estimate of V. In both cases, the underlying assumption is that the angular and positional errors are small enough to justify local linearization.

In practice, the position update is activated by the map whenever objects are predicted to appear in the current field of view of the sensor based on the estimated vehicle position. The error vector ΔV is initialized to 0 when the position update module is activated and is updated every time a new match is found between map and image objects. The module processes images until a sufficient number of matches is found, or until a maximum number of images, typically ten, is reached. The first stopping criterion is needed because a minimum number of matches is required for the final ΔV to be meaningful. The second criterion is needed because only the final estimate is sent to the other modules which might have to wait for a long time if there were no limit on the number of images processed, thus causing significant delays in the navigation system.

The output of this Kalman filter, ΔV and $C_{\Delta V}$, are position corrections based on landmark observations. They are converted to estimates of vehicle position P_m and its covariance C_m by adding the vehicle's position V and by dropping the rotation terms. The P_m and C_m terms are then fed to the Navigator module for its internal Kalman filter for overall vehicle position estimation, as described in Section 5.3.

5.2.2 Dead Reckoning

Measuring model for odometry

The odometer reports the vehicle's distance traveled. Since this data is vital to vehicle positioning and sensor modeling, a good model for the odometry system is essential.

There are two primary factors contributing to odometry data errors: terrain type and vehicle lateral acceleration (i.e. turning radius and velocity). The terrain type has a dramatic effect on the accuracy of an odometer due to slippage. For example, it is very easy to slip while traveling on loose gravel or mud versus asphalt. Slippage can also occur as a result of varying lateral acceleration. Driving though sharp curves at a constant velocity will cause more slippage than driving straight. For this study, we confined our driving to asphalt, so we modeled lateral acceleration slip but not slips induced by terrain type.

The odometer of our vehicle consists of a shaft encoder reporting data from the differential of the rear axle. In order to model this encoder data, experiments were conducted in a very large, flat, asphalt parking lot in order to keep the terrain type uniform and similar to typical situations. For each trial of the experiment, the vehicle was driven in a circle while the vehicle speed was kept as constant as possible (3-5 mph). The vehicle's real time controller kept the steering wheel at a constant position for each trial. Between successive runs, the steering wheel position was changed in small intervals in order to collect data across the full range of steering directions. At each steering wheel position, the vehicle was driven in a circle and the true circumference of the circle was measured by a fifth wheel measurement device pulled behind the vehicle. In cases where the circle was too large for the available space, the vehicle was driven

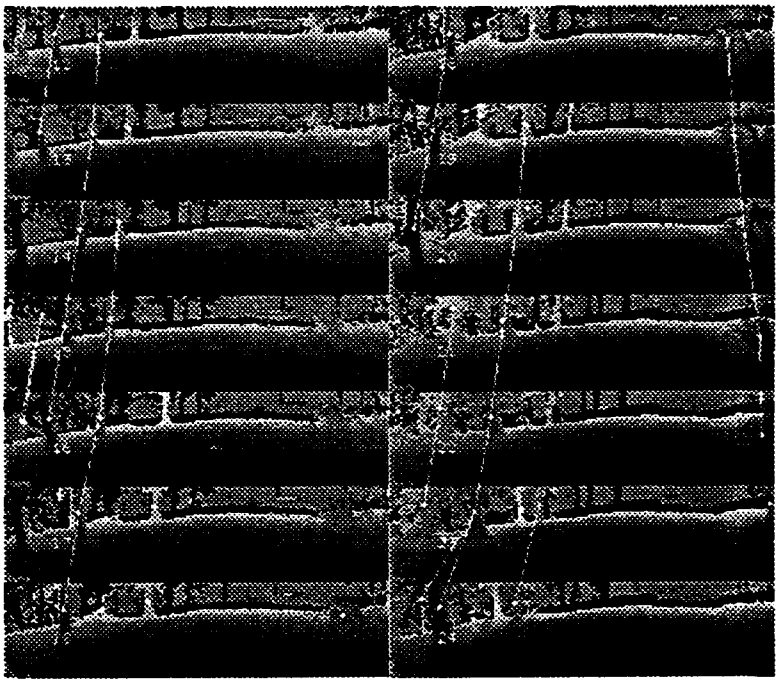


Figure 5.1: Fourteen ERIM laser range finder images. Dark pixels are close to the sensor and light pixels are farther away. Corresponding objects are linked together across images.

on smaller portions of a circle (half, quarter, etc). The data from the fifth wheel was recorded and averaged if multiple data was taken and used to compute the true distance traveled by the vehicle's origin (center of the rear axle). The number of encoder counts received for each trial was recorded as well as the steering wheel position. Figure 5.2 shows the result of this experiment.

The extremes of the x axis represent the sharpest turning radii of the vehicle where one would expect the most slippage to occur. This is exactly what occurs as illustrated in Figure 5.2 by the longer distances traveled per encoder count at the extremes. Also, as expected, the slippage diminishes as the radii become less sharp. Unfortunately, contrary to our expectations, the data is not symmetrical about zero curvature and it is difficult to fit curves or use tables for estimating an accurate odometry calibration function. One reason for this asymmetry may be uneven loading of the vehicle causing more slip while making right turns. This and other hypotheses must be further investigated by more data collection in the future. The nature of the current data made it impossible to accurately model the dependence of the odometry accuracy on turning radius. Instead, we simply used the mean and the variance of the data in our position estimation algorithm. Additional data was collected by driving the vehicle on a straight line between two carefully surveyed points twenty times in order to have a better estimate of the mean and the variance. The mean for distance traveled was 0.38885 mm/encoder count with a standard deviation of 0.0072785.

The vehicle's lateral position is also susceptible to slip. We discovered the extent of this slippage by measuring the lateral position error at the termination point of the completed circles. As expected, the error reduced with decreasing curvature. More data is required for determination of the exact relationship of the error to curvature. The mean (0) and the variance (0.51estimation.

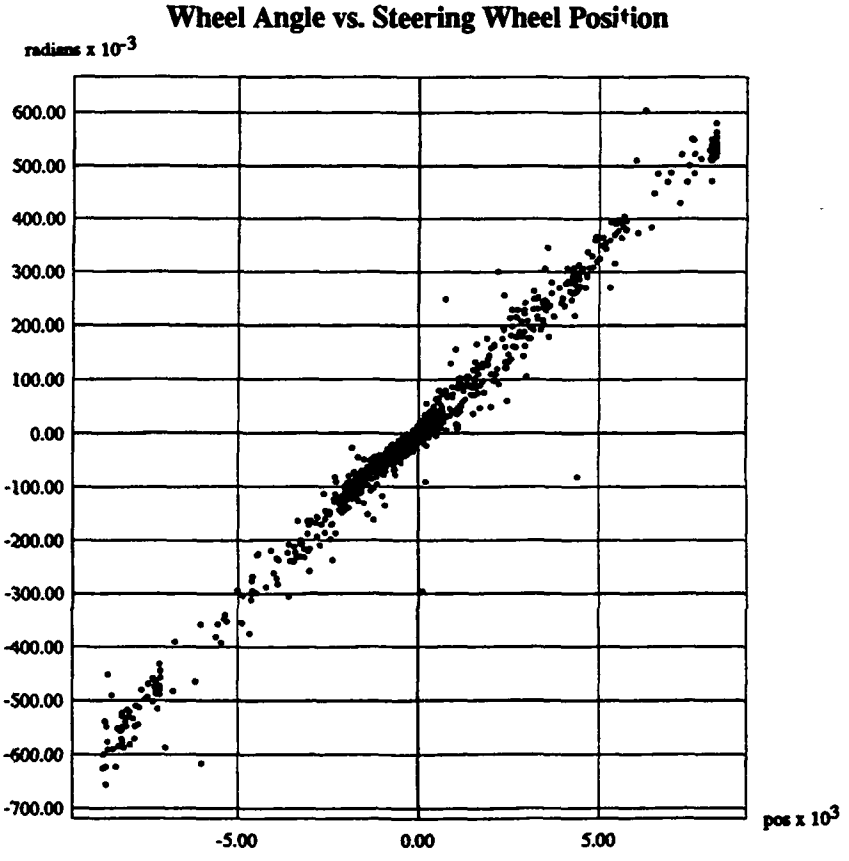


Figure 5.2: Noise in distance traveled and its dependence on vehicle steering. Vertical axis is calibration of distance traveled. Horizontal axis is steering wheel position from left turn to right turn.

Measuring model for steering angle

The turning radius of the vehicle is another crucial variable in position estimation. Estimating the turning radius requires accurate measurement of the effective front wheel angle α. The simplest way to measure α is to measure the position of the steering wheel. This requires a calibration function mapping the steering wheel position to α. To determine this function, the distance traveled was sampled from the odometer Δs and changes in heading $\Delta\varphi$ were recorded from a gyroscope at short time intervals as the vehicle was driven on a wide variety of paths at different speeds. The ratio of Δs to $\Delta\varphi$ is the sensed radius of curvature.

Provided that the interval is short enough, we can assume that the steering wheel position is essentially constant during this interval. Therefore, we can associate the measured radius of curvature with the particular steering wheel position. The sensed curvature was stored in a table indexed by the steering wheel position and vehicle speeds. For a better approximation of the steering wheel position, the steering encoder positions reported at the beginning and end of the time interval were averaged and used to index the curvature table. Different measured radii for the same steering encoder position were averaged as well. Figure 5.3 illustrates the data in these tables.

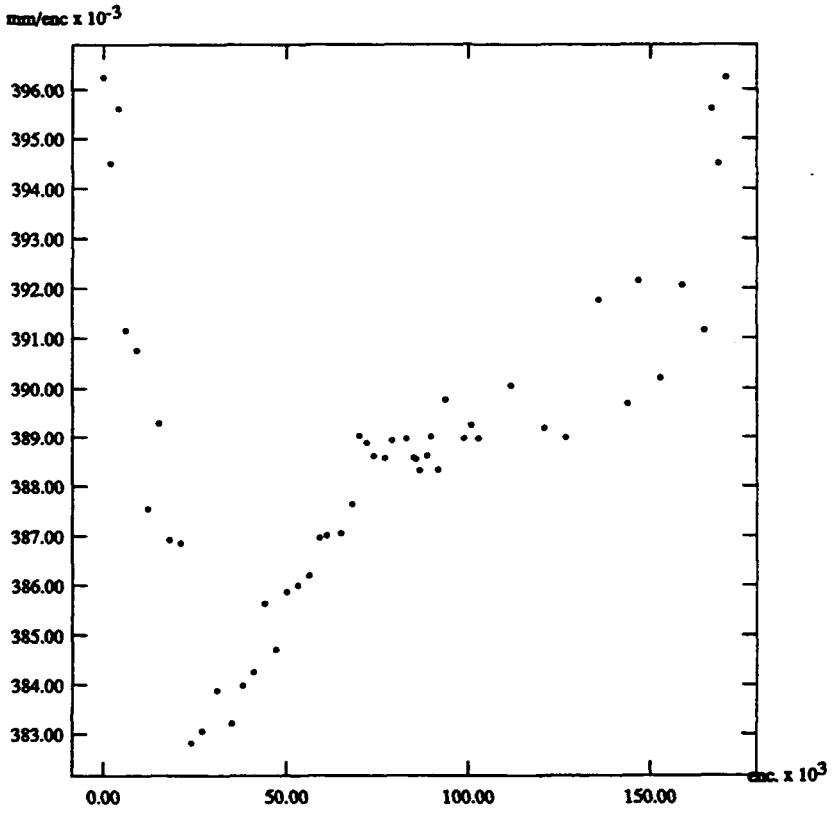


Figure 5.3: Wheel angle vs. steering wheel position

There are several possible reasons for the uncertainty in the estimated curvature from the steering wheel position. Tire slippage and deformation cause under/oversteering which varies the turning radius. Distortion in the sidewalls of the tires is considered to be a major contributor to these errors since it causes the tire treads to move in a different direction from the wheel rims. More detailed analysis of these factors and effects of vehicle dynamics are necessary to discover other sources for this uncertainty in the curvature estimation. Position estimation using the steering wheel and odometry data was quite accurate for distances less than twenty meters in typical situations. For longer distances, the heading error was significant and prompted the usage of more accurate sensors for vehicle heading. To reset the heading error, a gyroscope was used to measure vehicle heading and it was found to be extremely fast and accurate. Currently, the data from the gyroscope in conjunction with the odometry is used for position estimation.

5.2.3 Neural Network Road Following

We employ neural network based vision for the road following component of the system. A single hidden layer back-propagation network is trained to steer the vehicle using images from a video camera on board the Navlab. The video images are projected onto the low resolution "retina" which comprises the input layer of the network (See Figure 5.4).

Each unit in the input retina is fully connected to a layer of five hidden units which are in turn fully connected to a layer of 30 output units. The output layer is a linear representation of the direction the vehicle should travel in order to keep the vehicle on the road. The centermost output unit represented the "travel straight ahead" condition, while units to the left and right of center represented successively sharper left and right turns.

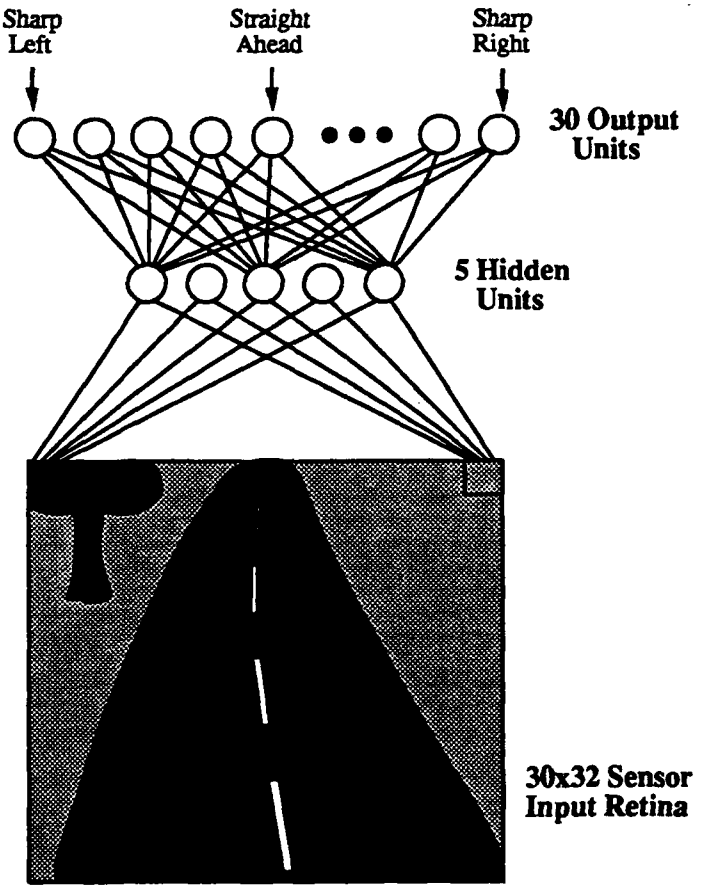


Figure 5.4: Architecture of the neural network vision system used for road following

To drive the Navlab, an image from the video camera is projected onto the input retina. In the forward propagation phase, the retinal image produces a pattern of activation on the hidden units, which then produces a pattern of activation on the output units. The influence one layer has on the next is determined by multiplying the activation level of the units in the first layer by the strengths of the connections between the two layers. After completing this forward propagation, a steering command is read off the output layer. The most active output unit determined the direction in which to steer the vehicle.

The network learns to drive by showing it images of the road ahead and training it to output the direction the human driver was steering in when each image was digitized. Using this technique, an individual network can learn to imitate the steering characteristics of a person on a particular road type in under 5 minutes. For a more detailed

description of neural network based vision for autonomous driving, see [8].

Uncertainty Modeling

In order refine the vehicle's estimated position using the output of the road following system, it is necessary to have an accurate model of uncertainty associated with the road follower's performance. Since the road following module is used to make lateral corrections to the estimated vehicle position relative to the road, the important aspect of its performance to quantify is how closely the road follower keeps the vehicle to the center of the road (or the center of the appropriate lane on a multi-lane road). More specifically, we must measure the mean and variance of the road follower's lateral error with respect to the road.

Measuring the road follower's accuracy is a two step process. The first step involves precisely determining the position of the road center. This measurement process is difficult on the single lane road used as the primary test site for this research, since broken pavement and fallen leaves frequently made estimating the exact location of road edges difficult. The technique chosen to ensure maximum measurement accuracy is manually locating the road edges and marking the point midway between them as the road center. To avoid disturbing the vision system with spurious image features, the road center points are marked with paint which contrasts only slightly with the pavement. We delineate the road center points once per meter over a 140 meter stretch of the test road, which included a straight stretch and turns to the left and right.

The second step in quantifying the driving accuracy of the road following module is to measure how closely the vision system kept the vehicle to the road center while driving autonomously. Because of the low contrast between the pavement and the road center markings, the only feasible technique for determining how close to the center the road follower keeps the vehicle is to measure it manually. Accurate real time measurement of the vehicle's relative position every meter proved impractical, since the vehicle's usual speed is at least two meters per second. So instead, we have designed an apparatus which allows the vehicle to leave a trail which can be carefully measured after the test run. Specifically, we immersed one end of a narrow tube in a bucket of water, and attached the other end to the origin of the vehicle's coordinate system, in the center of the rear axle. Using this apparatus as a siphon, the vehicle drips water along the precise path it follows over the test road. By measuring the deviation of this drip trail from the road center markings, we can estimate the mean and variance of the vision system's road following error. Since the trail is only water, it evaporates quickly. This prevents the trail from interfering with trails made on subsequent runs.

With the road follower driving at 5 miles per hour along the 140 meter test section of road, the vehicle's mean distance from the road center is 1.62 cm, with a variance of 52.49. These figures compare favorably with the same measurements made when a person drives the vehicle. Under human control, the vehicle's mean distance from the road center is 4.02 cm, with a variance of 30.03. It appears that the human driver, while more consistent than the vision system, had a inaccurate estimate of the vehicle's centerline, and therefore drove slightly right of the road center. Studies of human driving performance have found similar steady state errors and variances in vehicle lateral position. Blaauw [1] found consistent displacements of up to 7 cm were not uncommon when people drove on highways. On the same task, Blaauw reports standard deviations in lateral error ranging from 16.6 to 36.4 depending on the individual.

5.3 Annotated Maps

Once the individual components are built and calibrated, we still need a software framework to tie all the pieces into a coherent and useful system. We use the EDDIE architectural toolkit to combine information from the road follower, from the landmark recognizer, from dead reckoning, from the map, and from human planning in order to perform an autonomous mission. EDDIE consists of a communications library, which sets up direct communication links between processes, a database library, which maintains a local "list" of objects in the map, and a set of resources. These

resources are servers that clients can talk with via the communications library. The two most significant resources are the Annotated Maps module [9], which is used as both a means to efficiently query the object database and as a positional alarm clock, and the Navigator module, which is used to maintain the best estimate of the vehicle's position on the map.

The object database contains all the information about the world. The map can contain the following types of objects:

- Generic objects: Discrete circular or polygonal objects, e.g., trees, mailboxes.
- Road points: Road points are linked together to form roads
- Intersections: Intersections link different roads together
- Meta objects: Meta objects are used primarily for global information about roads. For example, a road's meta object might contain the name of the street.
- Alarms: These are artificial objects placed in the map for control purposes.

Each object in the database can be annotated with extra information that individual clients in the database have to interpret.

The Annotated Maps module takes the list of object descriptions and forms a two dimensional grid so it can answer queries about the object database more efficiently. For example, the object matcher has to know what objects are in its field of view. To find this out it could look through the object database, checking for objects that overlap the field of view polygon, but this would be very inefficient for a large database. Instead, the recognizer can send the polygon to the Annotated Maps module which can use the grid to quickly find the objects within the polygon.

The Annotated Maps module is not just a passive geometric database, it is actively involved in the control of the system. The alarm objects are placed in the map by a human user to plan the scenario. Alarms are conceptual objects in the map, and can be lines, circles, or regions. Each alarm is annotated with a list of client modules to notify and the information to send to each when the alarm is triggered. When the Annotated Map manager notices that the vehicle is crossing an alarm on the map, it sends the information to the pertinent modules. The map manager does not interpret the alarm annotations: that is up to the client modules.

Alarms can be thought of as positionally based production rules. Instead of using perception based production rules such as, "If A is observed, then perform action B", an Annotated Map based system has rules of the form, "If location A is reached, then perform action B". Thus we reduce the problem of making high level decisions from the difficult task of perceiving and reacting to external events to the relatively simple task of monitoring and updating the vehicle's position.

The first step in building an Annotated Map is collecting geometric information about the environment. We build our maps by driving the vehicle over roads and linking the road segments together at intersections. Once a road is built, the human user can create a "meta object" for that road which describes the whole road. In this system, we annotate a road's meta object with a short description or label and with the absolute orientation of the vehicle at the start of the road. At the same time, a laser range finder is used to record the positions of landmarks such as mailboxes, trees, and telephone poles. The landmark collector also annotates each object it finds with the two dimensional covariance of the object's position, in addition to its location and description, which can be used later by the object matcher. After the map building phase, a human planner draws "trigger lines" across the road at various locations. For example, the person knows that when approaching an intersection, the vehicle should slow down, and so chooses to put the trigger line at the approach to an intersection. The trigger line goes across the road at that point, and is annotated with a string of bits that represents the new speed of the vehicle. During the run, when the vehicle crosses the trigger line, the map

manager sends the string of bits to a module that interprets the information and slows the vehicle to the desired speed. In our current system, alarms are interpreted as commands, but there is no predefined "correct" way for a module to react to an alarm. Depending on its content, an alarm could also be interpreted as a wakeup call, or even as simply advice.

The EDDIE toolkit glues together our system, with the database providing the information, with the Navigator fusing position estimation's to pinpoint position, and with the Annotated Maps module providing the symbolic information and control knowledge necessary for a fully autonomous mission.

Because position information is so critical to an Annotated Map system, the Navigator resource module uses multiple knowledge sources to determine the vehicle's current location in the map. The Navigator uses the Kalman filter techniques described elsewhere in this paper to combine positioning information from our inertial navigation system, from the road follower, and from the landmark matcher to refine the vehicle's map position. The Navigator sends updates of this position to all modules that need global map positioning.

5.4 Filtered Position Updates

5.4.1 Corrections

In our implementation, the Navigator maintains a correction transform rather than the actual position. The vehicle controller provides positioning information completely independently of the map. This positioning information is derived from gyros and dead reckoning, and is in an arbitrary frame of reference that we call the controller coordinates. Client modules that do not need the map can use positioning information in controller coordinates. For example, a road following vision system might be interested only in the motion of the vehicle between frames, since it does not need to know the absolute position of the vehicle on the map, so it can directly use the controller position to get relative motion. Other modules, which need to use the map, need the transform T_{world} which converts controller coordinates to map coordinates. This T_{world} transform is maintained by the Navigator. Thus, a module which needs to know the current position of the vehicle on the map must first acquire the current T_{world}, then query the controller for the current vehicle position in controller coordinates, then apply T_{world} to convert to map coordinates.

There are several advantages of this system. First, the controller coordinates are never updated, so modules do not have to worry about jumps in relative locations of nearby objects that are stored in controller coordinates. In other schemes, when a position correction happens all objects stored in the current local coordinate system must be updated. Second, the position corrections which do occur (when the T_{world} is updated) can occur in a separate, possibly slower process (the Navigator), without interrupting the real-time loops of the controller. This decomposes our system logically into a fast system running in local coordinates, and a separate set of often slower processes that need to refer to the map. Besides being a good design tool, this is also a practical help in doing the updates. Time is no longer as important a quantity. Directly updating the controller's position would require changing the position estimate in real time, while the controller is still doing dead reckoning. In our scheme, instead, the Navigator's T_{world} can be updated with no real-time concern, since it is not rapidly changing. Also, filtering the relatively stable quantity T_{world} is relatively easy since it will typically require only small changes. This also means that the compounding operation, which increases positional uncertainty during dead reckoning between landmark fixes, is also simplified. The mean of the T_{world} stays constant during compounding, and only the positional covariance C_p must be changed.

5.4.2 Filter Design

To estimate the vehicle's position as accurately as possible, odometry data, gyroscope heading data, ERIM data, and road information are used in conjunction with previously recorded maps to estimate vehicle position and uncertainty.

The vehicle heading is measured by a commercial gyroscope which employs its own Kalman filters to encorporate compass and gyroscope data to produce exceptionally accurate heading data. This gyroscope was made by the Litton corporation for aiming guns accurately. For this reason, the filter formulation is only concerned with the vehicle's x and y coordinates assuming perfect heading (theta) estimation from the commercial gyroscope.

The Navigator uses the real time controller's position estimation derived from the odometry and heading data as a basis for an internal transform developed from ERIM and map information. The Kalman filter is a black box to the Navigator which internally maintains the position uncertainty. The Navigator invokes two operations in this formulation, the compounding and the merging. Figure 5.5 shows the covariance compounding and merging equations.

Compounding Equations		Kalman Filter Equations		
$C'_P = J\begin{bmatrix} C_P & 0 \\ 0 & C_V \end{bmatrix} J^T$		$C'_P = (C_P^{-1} + C_M^{-1})^{-1}$ $\bar{P}' = \bar{P} + K(\bar{P}_M - \bar{P}')$		
$C_P = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{bmatrix}$	$C_D = \begin{bmatrix} \sigma_l^2 & 0 \\ 0 & \sigma_d^2 \end{bmatrix}$	$\bar{P} = \begin{bmatrix} x \\ y \end{bmatrix}$	$\bar{P}_M = \begin{bmatrix} x_E \\ y_E \end{bmatrix}$	$\bar{P}_M = \begin{bmatrix} x_R \\ y_R \end{bmatrix}$
$J = \begin{bmatrix} 1 & 0 & \cos(\theta) & -\sin(\theta) \\ 0 & 1 & \sin(\theta) & \cos(\theta) \end{bmatrix}$		$K = C_P(C_P + C_M)^{-1}$		

Figure 5.5: Kalman filter equations

The compounding operation is invoked at regular distance intervals of 0.5 meters. The covariance matrix for dead reckoning, C_D, contains the variance of lateral error σ_l and distance traveled σ_d. These variances are assumed independent. σ_l and σ_d are updated using the variances previously determined from odometry data and road follower performance. The new position covariance matrix C'_p is computed by using the Jacobian J to compound C_D and the old C_p.

The merging operation is invoked when another module provides an estimate of the vehicle position. In our current system, these additional estimates occur when the object matcher finds a new vehicle position from landmarks or when the road follower provides a steering command. The Navigator is given a vehicle position P_m measured by one of the perception systems and corresponding covariance matrix C_m. P_m and C_m are merged with the current estimates of position P and uncertainty C_p by the Kalman filter. This merging operation updates the covariance, C'_p, and returns the updated position P'. The Navigator uses the new map position P' and the controller position (x_c, y_c) to generate a new T_{world}.

The landmark matcher directly gives the Navigator a new estimated vehicle position P_m and its covariance C_m. The Kalman filter module directly uses these values, plus the old position estimate P, as the inputs to its filter.

To update P_m and C_m from the map from road information, the Navigator follows a similar procedure. First, the Navigator uses the current transform T_{world} to estimate vehicle position P. It then projects P onto the road, to find the

measured vehicle position P_m, since the road follower is assumed to keep the vehicle on the road center. Road updates can move the vehicle position perpendicularly to the road but not along the road. The Navigator therefore forms a covariance matrix C_m oriented along the road, with the road follower's error estimate perpendicular to the road and an essentially infinite variance along the road. As in landmark matching, the P_m estimate, the C_m covariance, and the P prior position estimate are passed to the Kalman filter module to generate a new P' and C_p. The new P' is used by the Navigator to build its new T_{world}.

5.5 Experiments and Results

5.5.1 Results

We tested the system on a twisting park trail near campus. A human driver drove up the road to map it while the object detector found landmarks such as large trees. The object detector annotated each landmark with its observed covariance in x and y. After we built the map, we put one trigger line in it which tells the landmark matcher to start looking for landmarks during a run. One such map is shown in Figure 5.6.

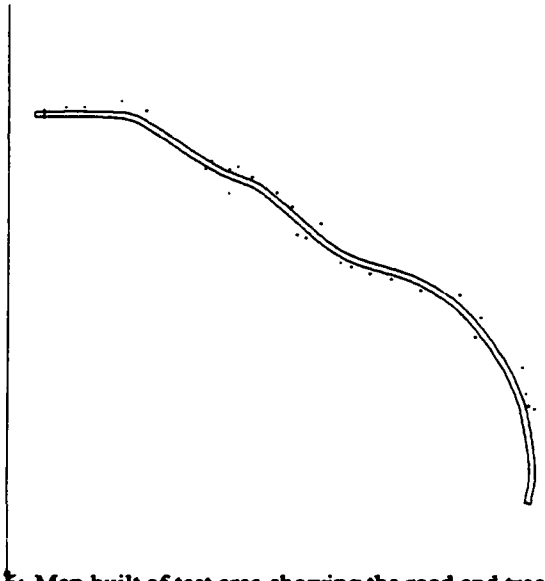


Figure 5.6: Map built of test area showing the road and trees.

To start a run, the driver drives the vehicle onto the road, and the operator then estimates the vehicle's initial position. The initial orientation is calculated by taking the difference between the angle currently reported by the gyros and the gyro angle annotated in the nearest road and adding that difference to the map angle annotated in the nearest road. The amount of drift in the gyros was small enough that this method of orienting the vehicle was much more accurate than just guessing the vehicles orientation. We estimated the initial uncertainty of the vehicles position as three meters in x and y. Figure 5.7 shows the uncertainty ellipse of the vehicle in the middle of the run after being

shortened along the direction of the road by landmark matching and after being thinned by updates from the road follower.

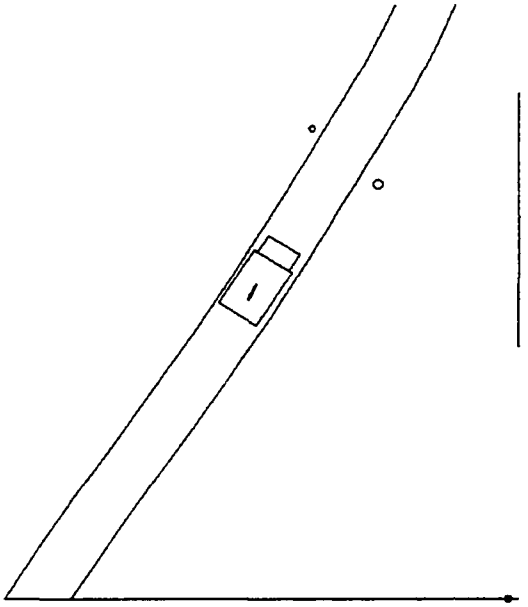


Figure 5.7: Vehicle position and uncertainty during a run

One interesting result is that making lateral road corrections on a curving road can correct an error in position along the road. This is not immediately obvious intuitively since an individual road correction gives no information about the position of the vehicle along the road, only information about the lateral position of the vehicle on the road. We first noticed this effect by observing the changes in uncertainty ellipses during a run. In Figure 5.8, a vehicle has a circular uncertainty at position a, and after a road update at position b the uncertainty ellipse becomes a needle in the direction of the road. Imagine that the vehicle travels through the corner. At position c it has an uncertainty ellipse very similar to at position b, but after another road update at position d, the ellipse shrinks drastically. Simply from local road updates, we are vastly more confident about the position of the vehicle along the road at position d than at position a. This implies that the apparently difficult problem of using road curvatures as landmarks is solved by simply making local corrections to the vehicle's position perpendicular to the road.

This effect was evident in our runs. We deliberately started the vehicle in the wrong place in the map, displaced along the road by 4 meters. The landmark matcher could not match landmarks with this large an error, so it failed initially. After going through a curve with just road updates, the position in the map was close enough to the position in the real world that the landmark matcher was able to match sensed objects with map objects and further refine the position.

Are there cases in which the road updates confuse the position along the road? Unfortunately, yes. The autonomous road follower drives slightly differently than the human driver who made the map. For instance, the road follower may cut a corner slightly more than the human driver. Our model assumes that the vehicle is in the center of the road, even though it is not. A series of very slightly erroneous corrections going through a corner will result in an error in the distance along the map. Figure 5.9 illustrates the problem. This problem could be addressed by having the road follower be the driver when making the maps, or by increasing the compounding error in the direction of travel when going through a curve, or by simply not performing road updates when the vehicle is going through a curve.

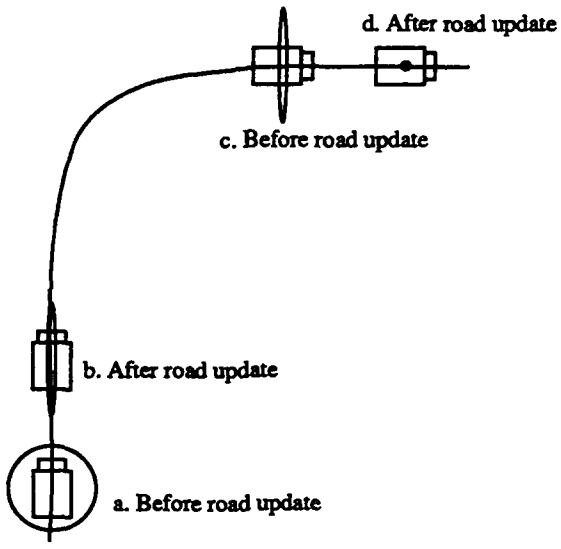


Figure 5.8: Effects of road updates on position uncertainty

We made a series of runs with the vehicle and measured performance qualitatively, and derived some preliminary quantitative measurements from system performance. Qualitatively, the most important criterion is that the landmark matcher should be able to find landmarks within its expected error. We found that to be the true in nearly all cases. Quantitatively, at the end of typical runs of 350 meters, the position updates generated by the matcher were usually between 5 to 10 cm.

5.5.2 Future Work

There are several areas still open for investigation.

Curves and matching. Since the road follower provides position updates only perpendicular to the road, we rely on knowing the road direction fairly accurately. If the vehicle is in a tight curve, and has significant positional error along the road, it will also have significant uncertainty in the direction along which road updates should take place. We need to model and account for that error, or to not perform road updates while driving on tight curves.

Map errors. Our error formulations does not account for errors in road position on the map. We do record the covariance of the estimate of landmark positions, but that is relative to an assumed perfect vehicle position on the road. A fairly accurate approximation is that the map has the same error characteristics as the vehicle driving, so treating the map as perfect and doubling the error estimates during retraversal should work. Better estimates of map error will become more important as we continue to work on building more accurate maps.

Correlated errors. We treat our errors as zero-mean, uncorrelated, with a Gaussian distribution. In fact, errors in certain systems like the road follower tend to be correlated in time: if the vehicle is 10 cm off at one instant, it is unlikely to be 10 cm the opposite direction in the next instant. This undoubtedly limits our performance.

Dead reckoning errors. We know that errors in dead reckoning are linked to radius of curvature. We believe that lateral and longitudinal errors are also linked. Our current data sets do not give us a meaningful basis for representing

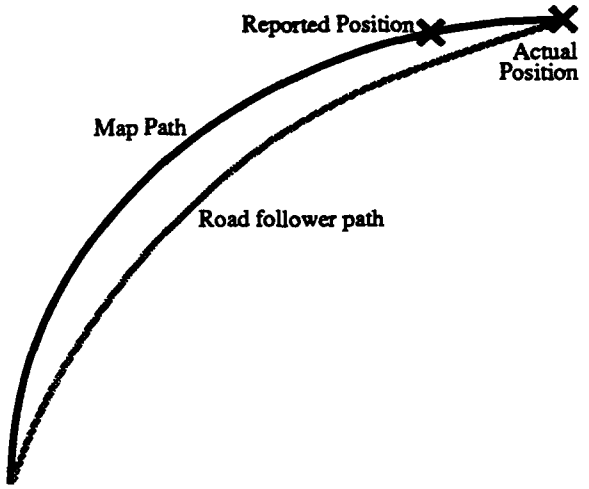


Figure 5.9: Errors caused by road updates during inaccurate driving through a turn.

those relationships. The net effect is to generate another source of correlated errors. If the vehicle drives along a long curve with constant radius, the mean distance traveled will have a small bias from the mean over all possible directions of travel.

Measuring system performance. It is comparatively easy to measure system repeatability, by making several runs over the same route and recording the end position of the vehicle. It is much harder to measure absolute accuracy. Our current test course is lined with trees, so there is no clear line of sight along which to use a theodolite. We will either have to subcontract a surveyor to make a complete map, or run in a less interesting test site just to make measurements.

Other environments. The tests described in this paper involve road following with relatively discrete landmarks, such as trees. The precision of our dead reckoning will degrade when we make cross-country runs on dirt and mud, and the precision of landmark matching will be less when we are trying to recognize less structured features such as piles of rocks.

5.6 References

[1] G. J. Blaauw. Driving Experience and Task Demands in Simulator and Instrumented Car: A Validation Study. *Human Factors*, 24:473–486, 1982.

[2] A. Gelb. *Applied Optimal Estimation*. MIT Press, 1974.

[3] M. Hebert. Building and Navigating Maps of Road Scenes Using Active Range and Reflectance Data. In Charles E. Thorpe, editor, *Vision and Navigation: The Carnegie Mellon Navlab*. Kluwer Academic Publishers, 1990.

[4] M. Hebert. Building Object and Terrain Representations for an Autonomous Vehicle. In *Proc. American Control Conference*, June 1991.

[5] M. Hebert and T. Kanade. Outdoor Scene Analysis Using Range Data. In *Proc. of Intern. Conf. on Robotics and Automation*, pages 1426–1432, San Francisco, April 1986. IEEE Computer Society.

[6] M. Hebert and E. Krotkov. Imaging Laser Radars. In *Proc. IROS 91*, 1991.

[7] M. Hebert, I. Kweon, and T Kanade. 3-D Vision Techniques for Autonomous Vehicles. In Charles E. Thorpe, editor, *Vision and Navigation: The Carnegie Mellon Navlab*. Kluwer Academic Publishers, 1990.

[8] D. Pomerleau, J. Gowdy, and C. Thorpe. Combining Artificial Neural Networks and Symbolic Processing for Autonomous Robot Guidance. *Journal of Engineering Applications of Artificial Intelligence*, 1991.

[9] C. Thorpe and J. Gowdy. Annotated Maps for Autonomous Land Vehicles. In *Proceedings of DARPA Image Understanding Workshop*, Pittsburgh PA, September 1990.